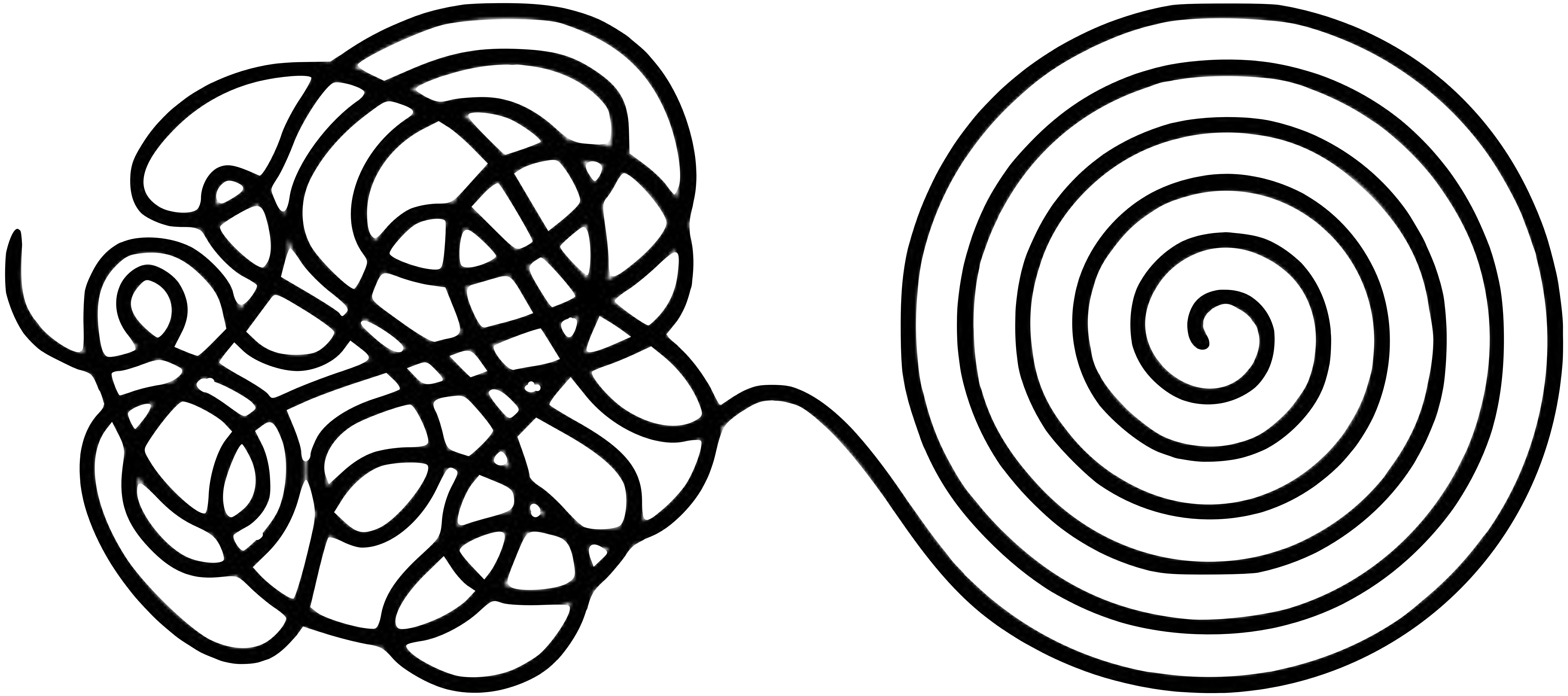
Declarative Process Management and Mining

A killer application for LTLf



Marco Montali
Free University of Bozen-Bolzano

LTLf SSS 2023, San Francisco

What do we do in Bolzano

We develop **foundational** and **applied techniques**

grounded in **artificial intelligence**, **logics**, **formal methods**, and **data science**

to create **intelligent agents** and **information systems** that combine

processes and **data**.

Three interconnected lines of research

AI for…

… flexible, declarative agents/processes

… process mining and operational support

… data-aware agents/processes

Advancement in the foundations and applications of:

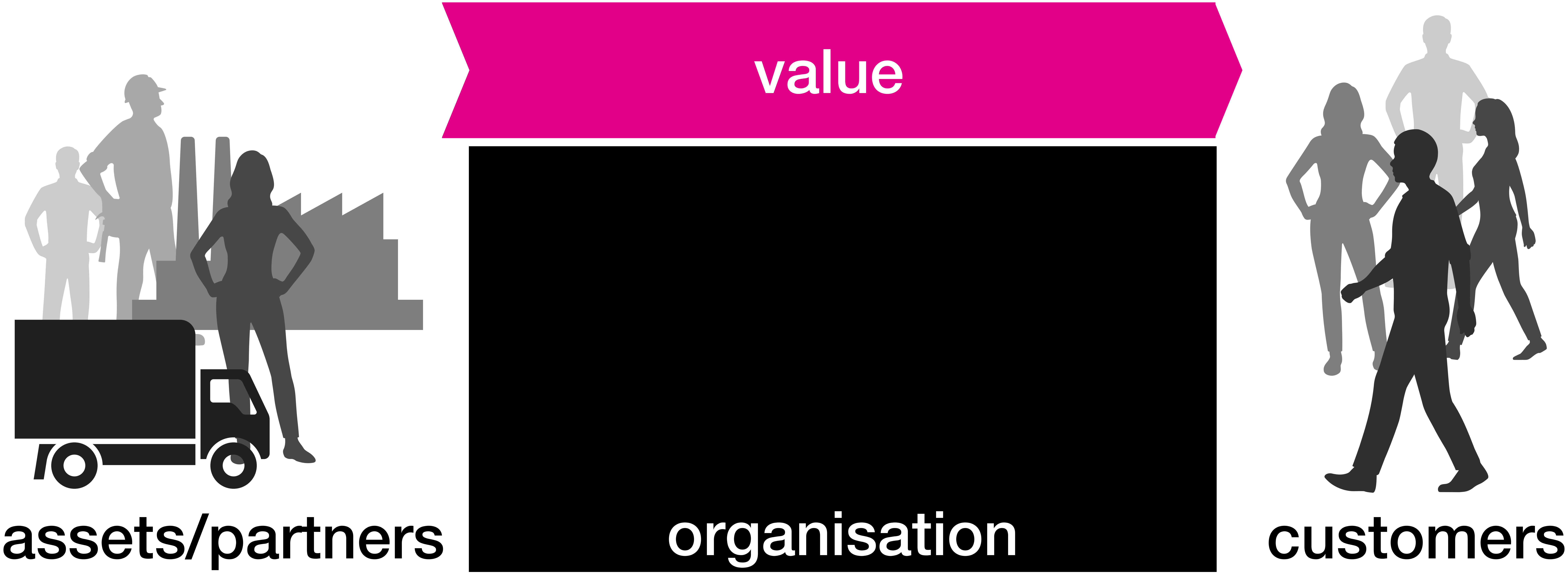
information systems

🔄

artificial intelligence

Warm up: process management and mining

Business process



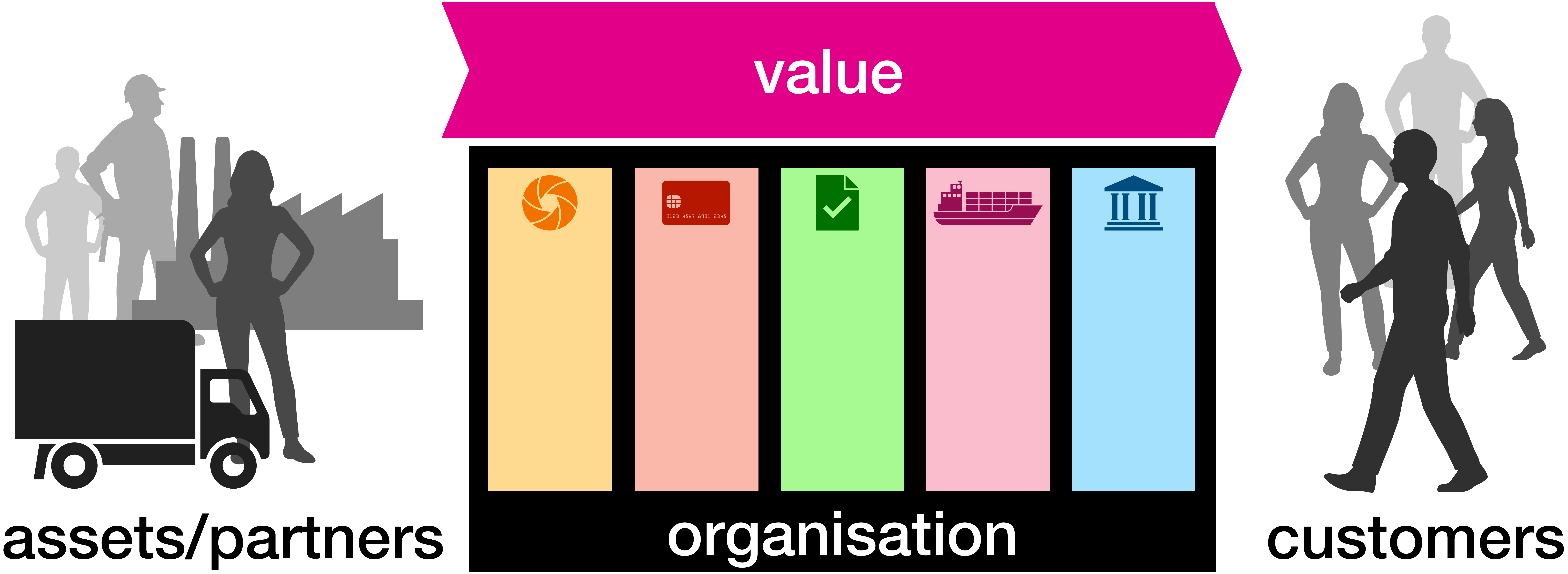
value

assets/partners

organisation

customers

Business process



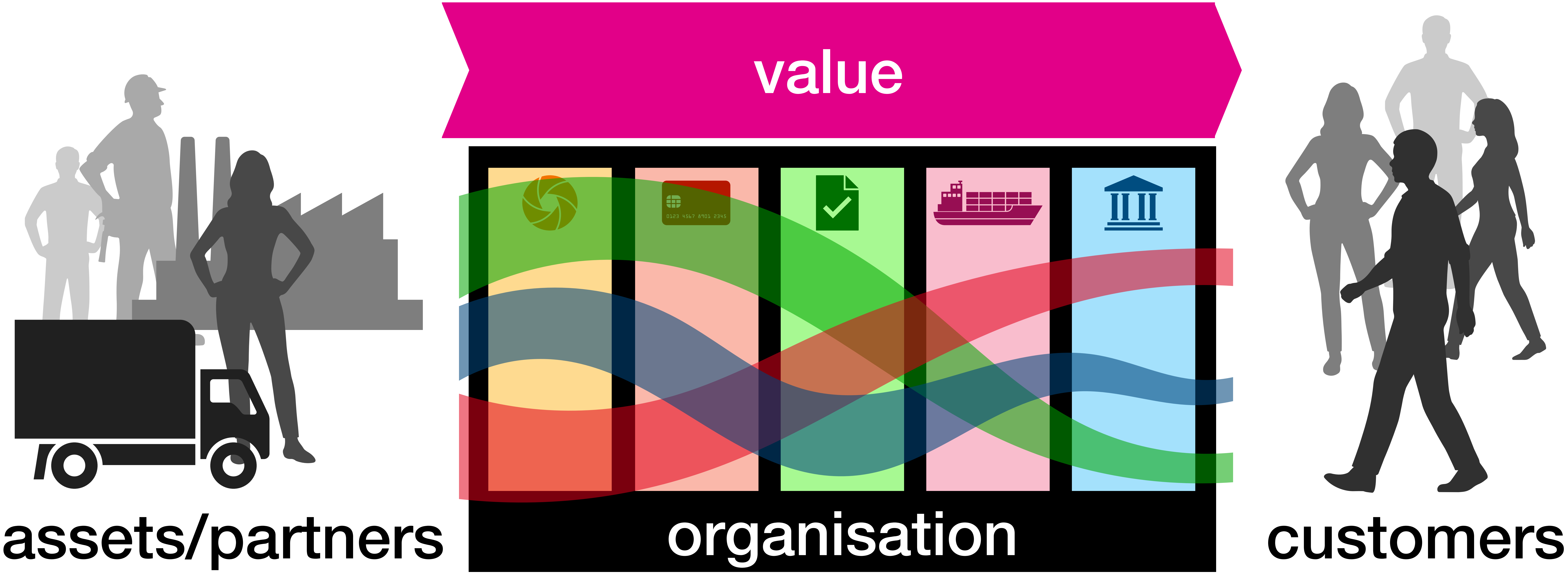
value

assets/partners

organisation

customers

Business process



A **business process** is a collection of **related events, activities** and **decisions**, that involve a number of ***actors and objects***, and that *collectively lead to an outcome that is of* ***value to an organization or its customers***

REALITY
process discovery
(re) design
configure/ deploy
enact/ monitor
data
models
IT support
knowledge workers
managers/ analysts
ORGANISATION

REALITY

process discovery

process models

data

enact/ monitor

(re) design

models

event logs

knowledge workers

managers/ analysts

IT support

configure/ deploy

ORGANISATION

ship id
inspect ship
record cargo residuals
cargo residuals
enter
receive entrance request
record ship info
decice clearance
acquire certificate
record exp. date
certificate exp. date
refuel
ship type (short name)

Case id	Event id	Properties				
		Timestamp	Activity	Resource	Cost	...
1	35654423	30-12-2010:11.02	register request	Pete	50	...
	35654424	31-12-2010:10.06	examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	check ticket	Mike	100	...
	35654426	06-01-2011:11.18	decide	Sara	200	...
	35654427	07-01-2011:14.24	reject request	Pete	200	...
2	35654483	30-12-2010:11.32	register request	Mike	50	...
	35654485	30-12-2010:12.12	check ticket	Mike	100	...
	35654487	30-12-2010:14.16	examine casually	Pete	400	

REALITY

enact/
monito

knowledge
workers

(re)
design

managers/
analysts

ORGANISATION

process models

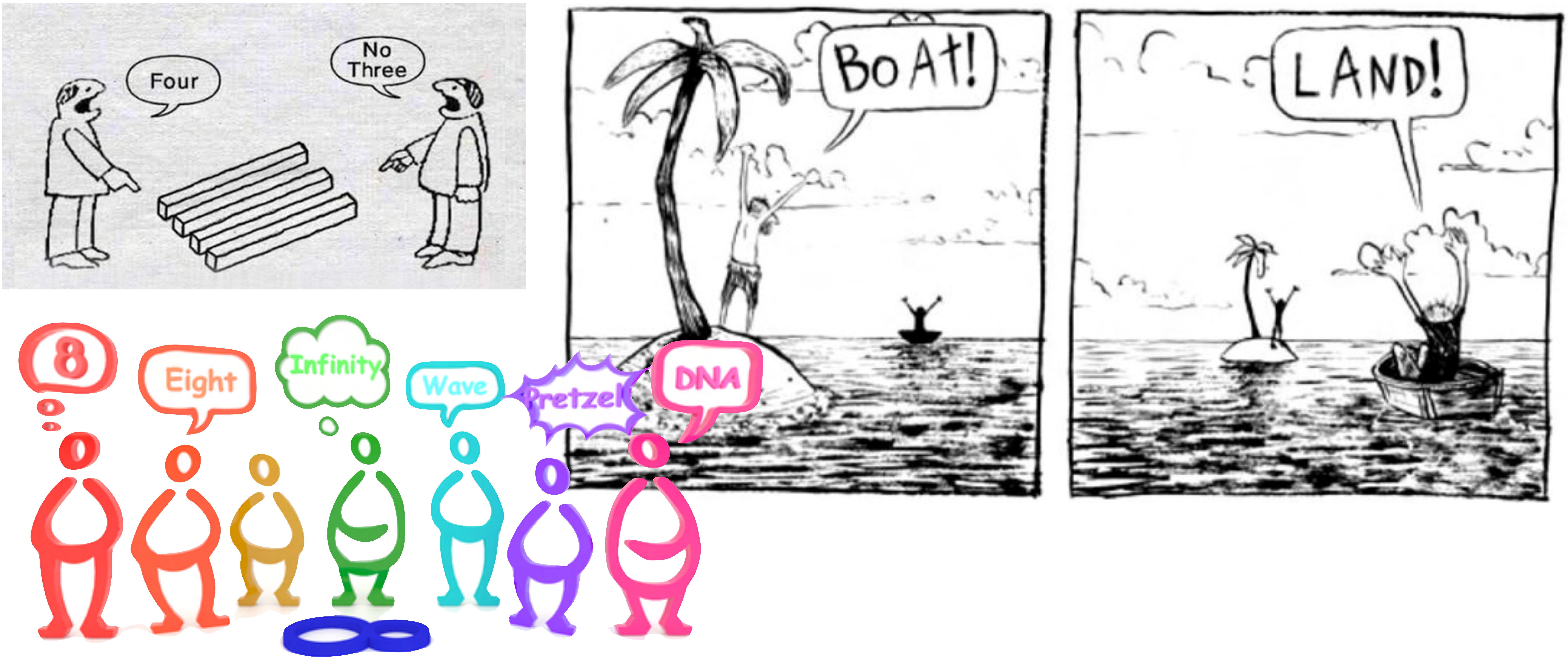
event logs

model-driven
process management

REALITY
process models
event logs
enact/
monito
knowledge
workers
(re)
design
managers/
analysts
ORGA ATION

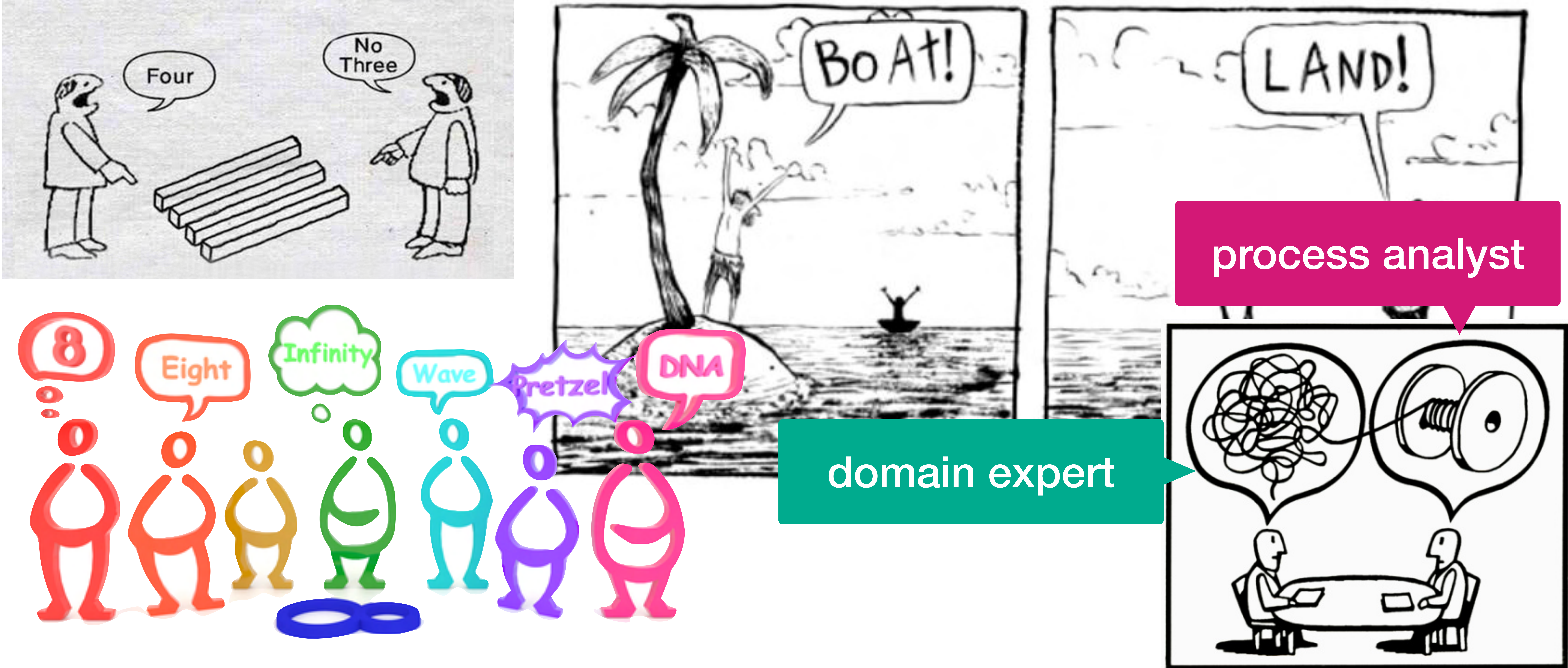
Model-driven process management

1. process discovery via conceptual modelling



Model-driven process management

1. process discovery via conceptual modelling

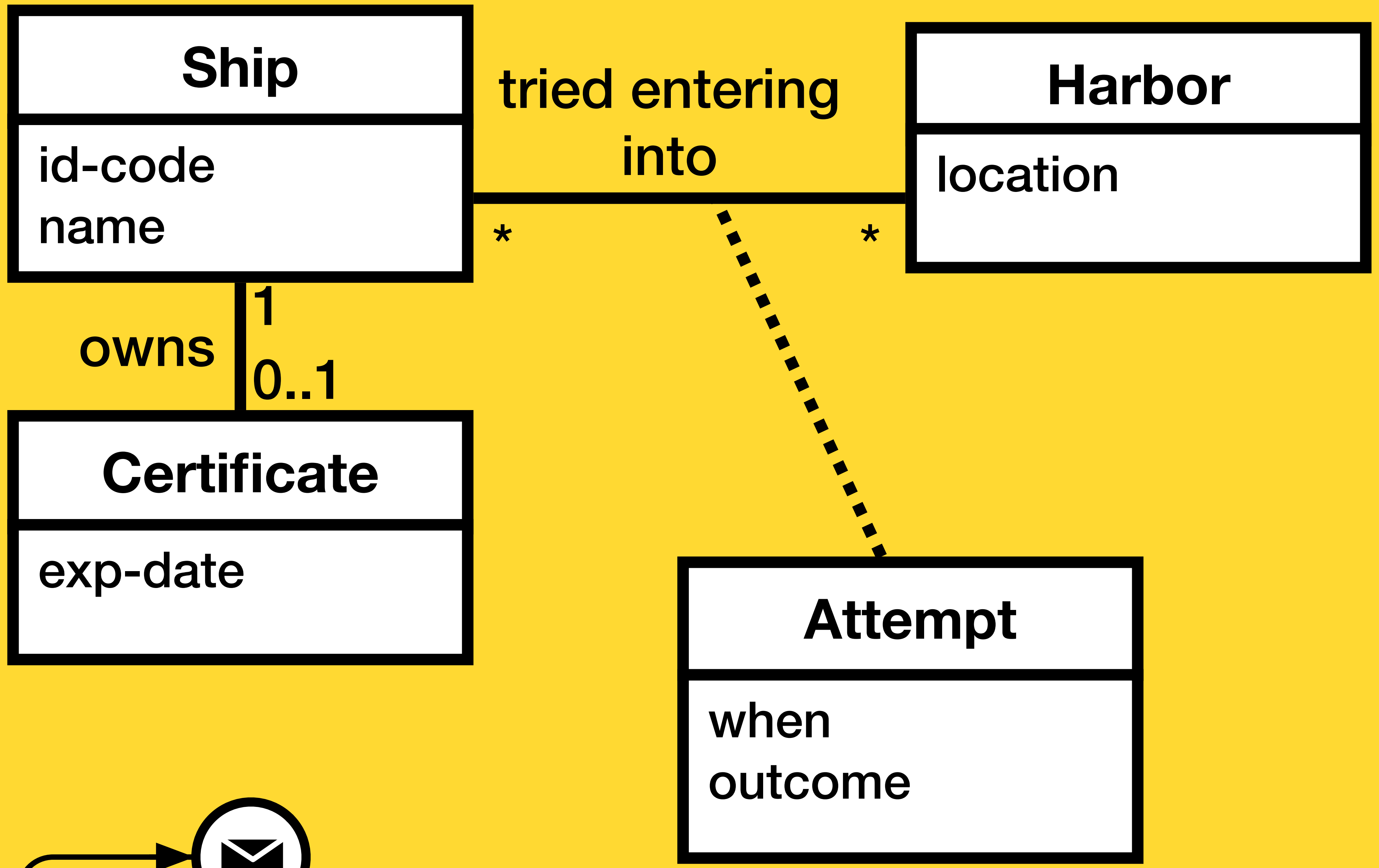


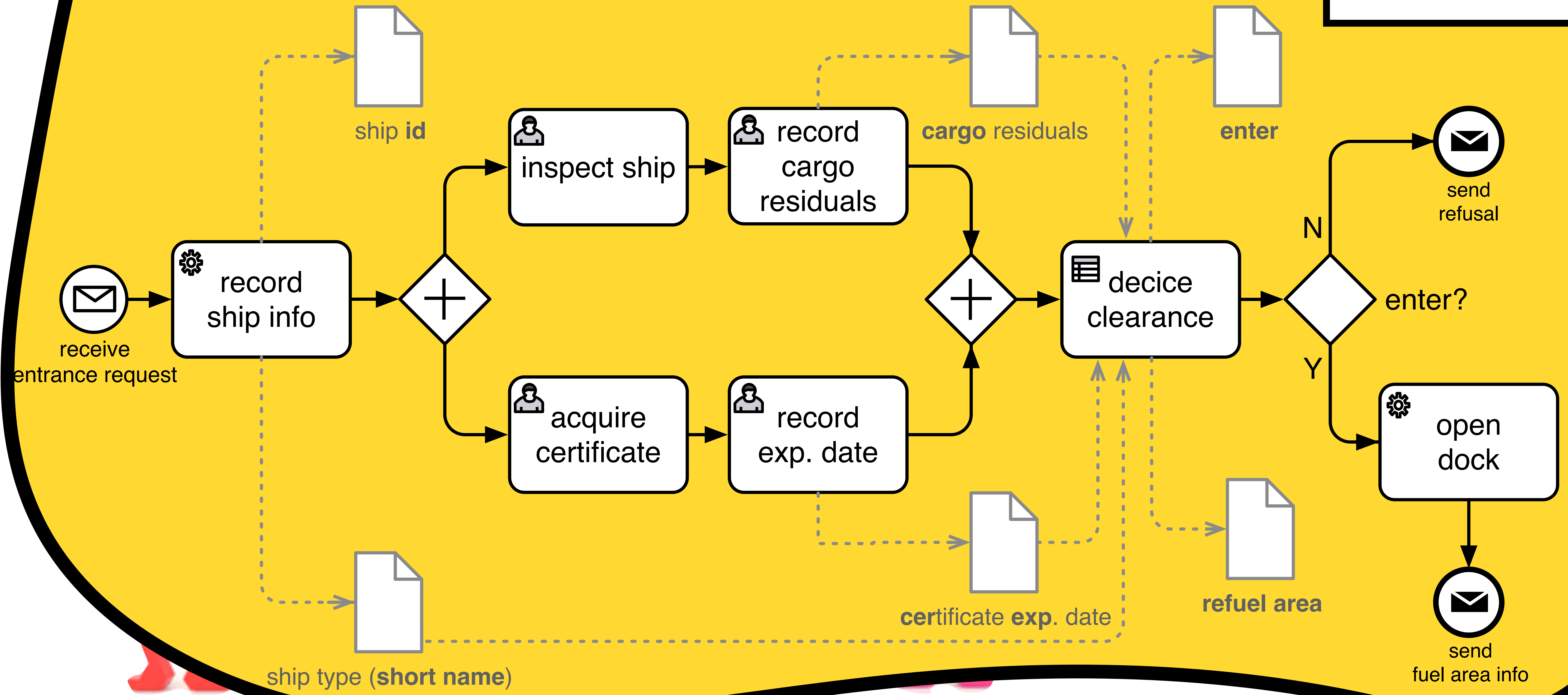
process analyst

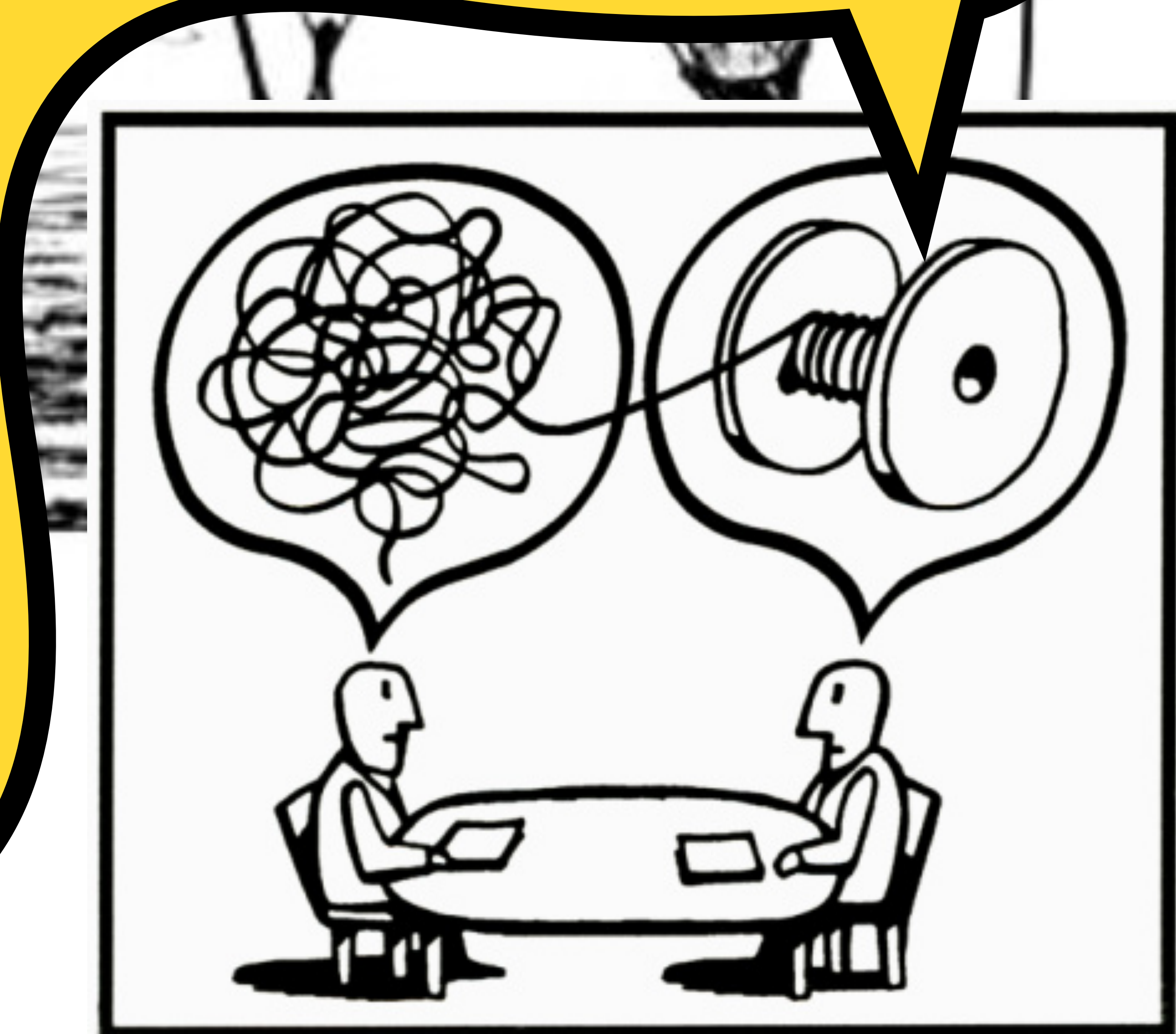
domain expert

Refuel area determination

U	Enter	Length (m)	Cargo (mg/cm^2)	Refuel Area
	y,n	≥ 0	≥ 0	none, indoor, outdoor
1	n	–	–	none
2	y	≤ 350	–	indoor
3	y	> 350	≤ 0.3	indoor
4	y	> 350	> 0.3	outdoor

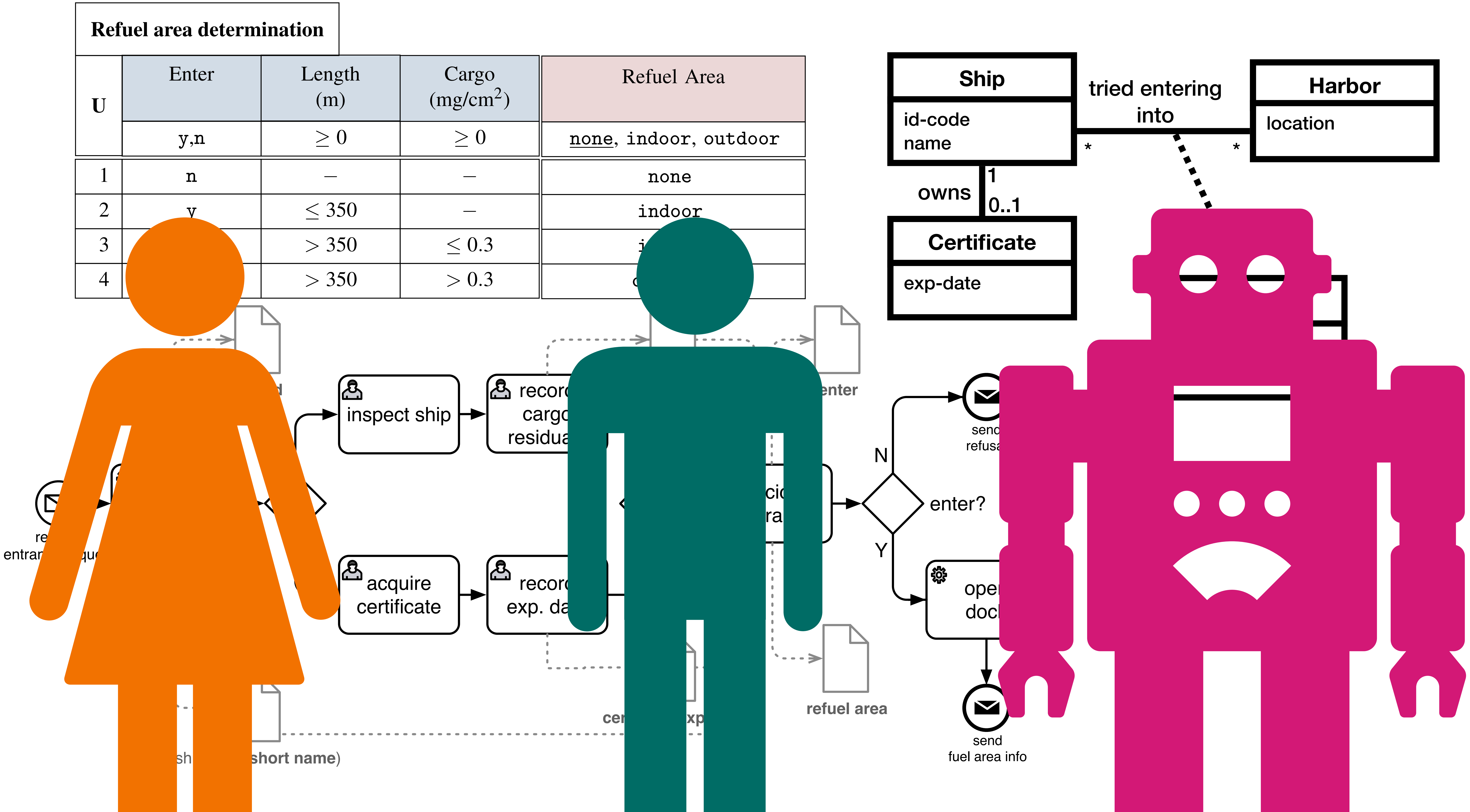






Model-driven process management

2. share and understand



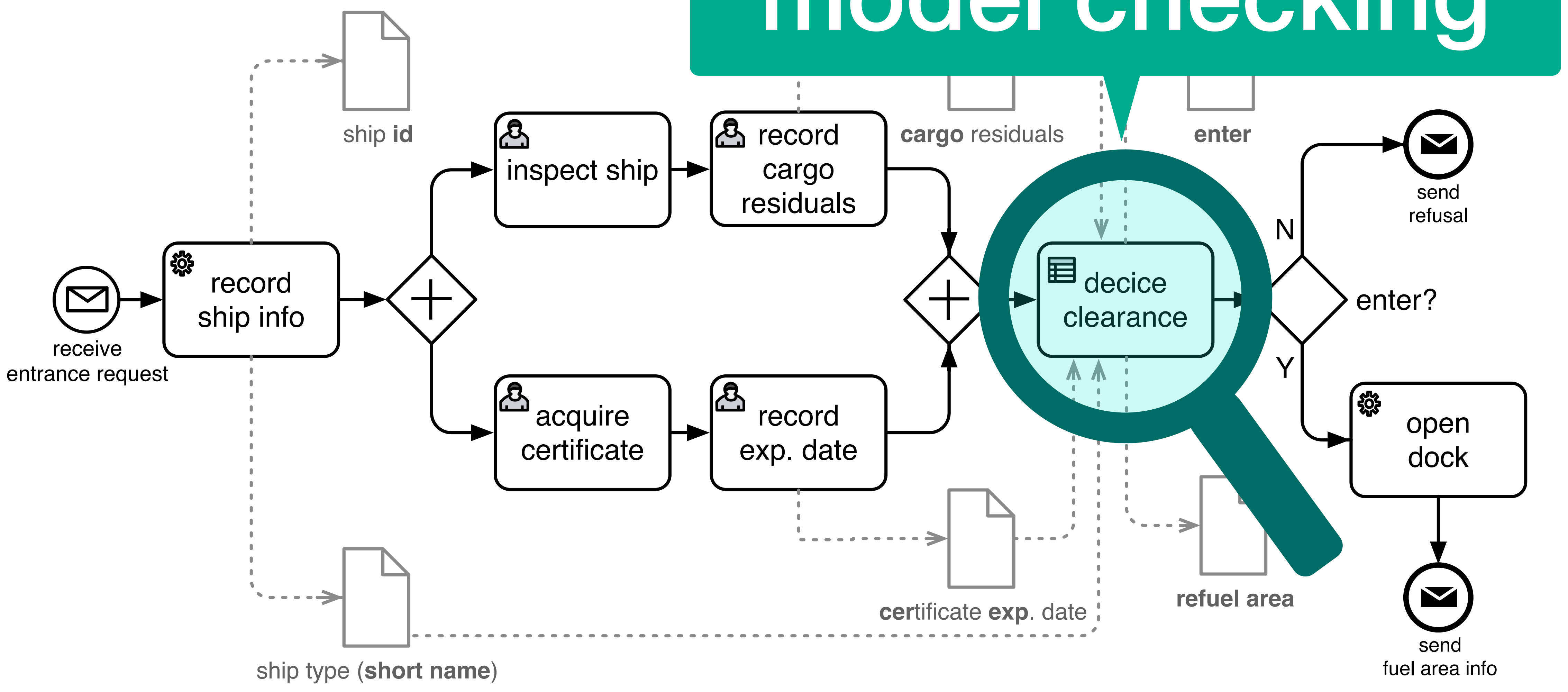
Refuel area determination

U	Enter	Length (m)	Cargo $(\mathrm{mg/cm}^2)$	Refuel Area
	y,n	≥ 0	≥ 0	none, indoor, outdoor
1	n	–	–	none
2	y	≤ 350	–	indoor
3		> 350	≤ 0.3	i
4		> 350	> 0.3	o

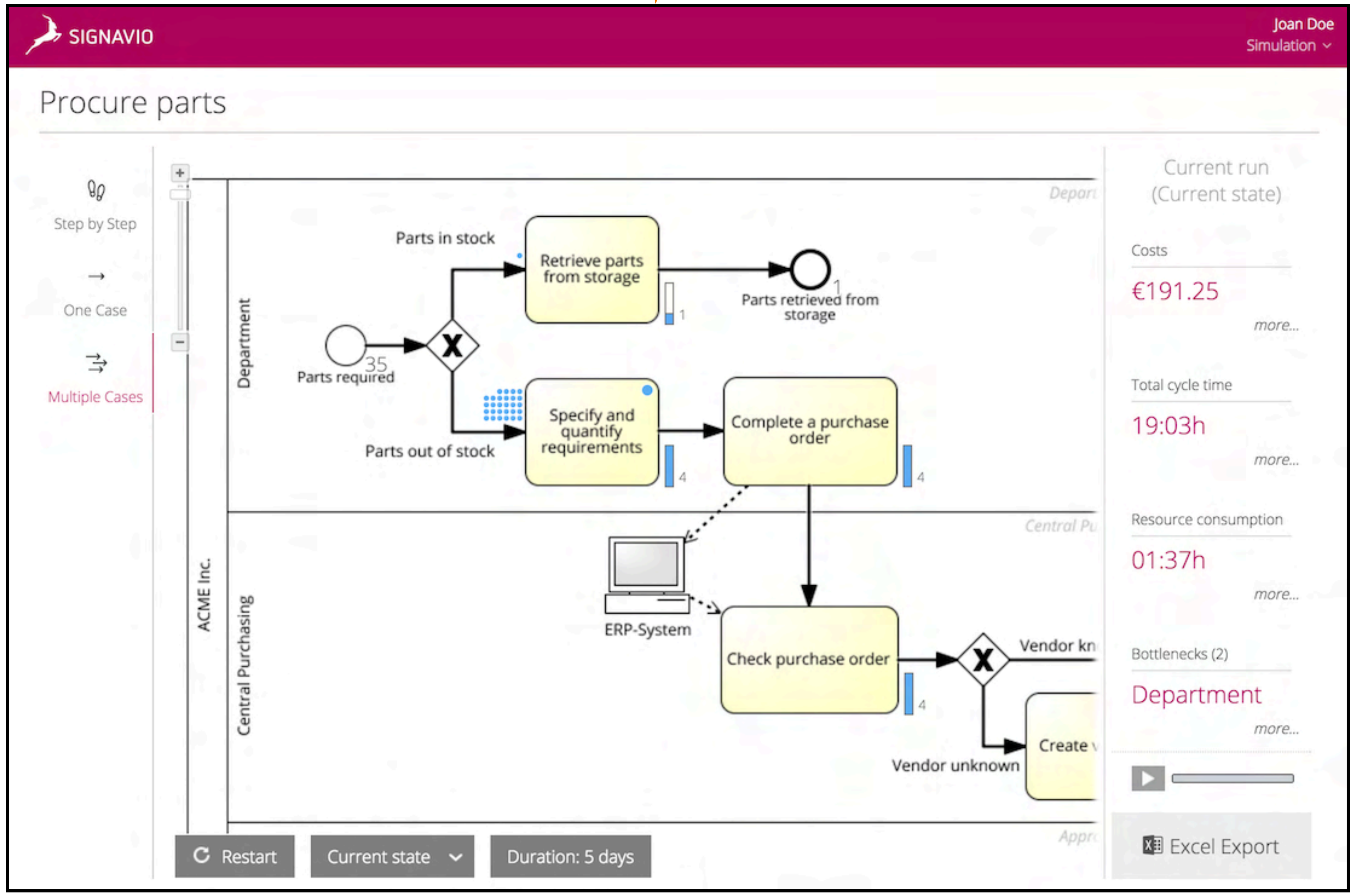
Model-driven process management

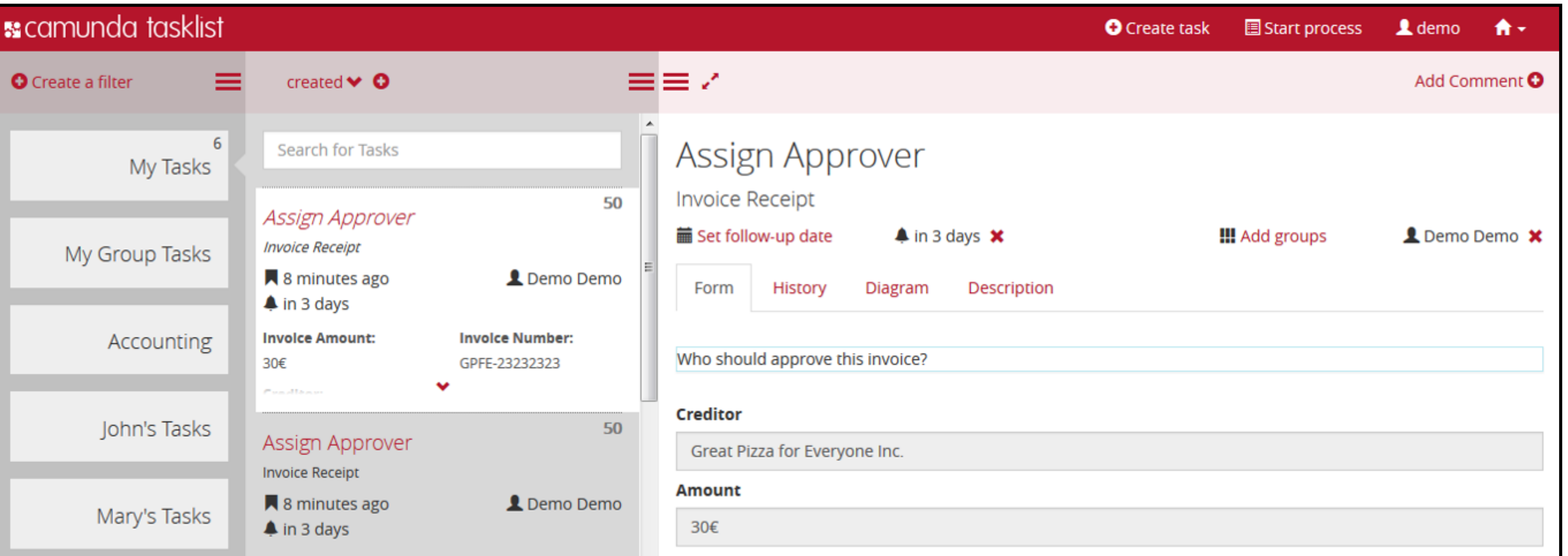
3. use models

model checking



quantitative analysis/simulation





Assign Approver 50
Invoice Receipt
8 minutes ago — Demo Demo
in 3 days
Invoice Amount: 30€
Invoice Number: GPFE-23232323

Assign Approver 50
Invoice Receipt
8 minutes ago — Demo Demo
in 3 days

Assign Approver
Invoice Receipt
Set follow-up date — in 3 days — Add groups — Demo Demo

Form | History | Diagram | Description

Who should approve this invoice?

Creditor: Great Pizza for Everyone Inc.

Amount: 30€

execution support

REALITY

knowledge workers

enact/ monito

event logs

process models

(re) design

managers/ analysts

ORGANISATION

REAL
process models
event logs
enact/
monito
(re)
design
knowled
workers
managers/
analysts
ORGANISATION

data-driven
process management

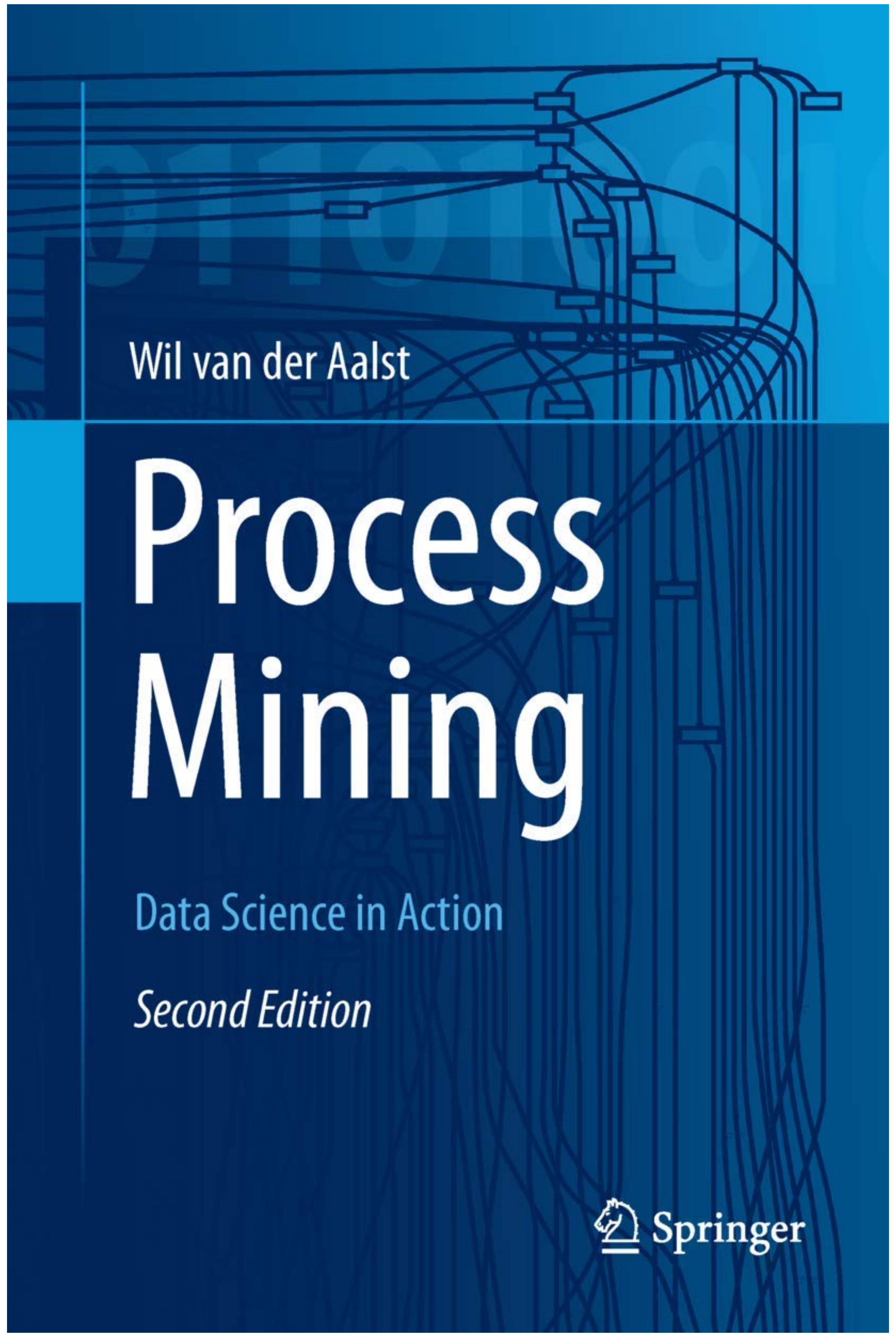
Management vs reality

process model

Management vs reality

real process

Process mining: “data science in action”



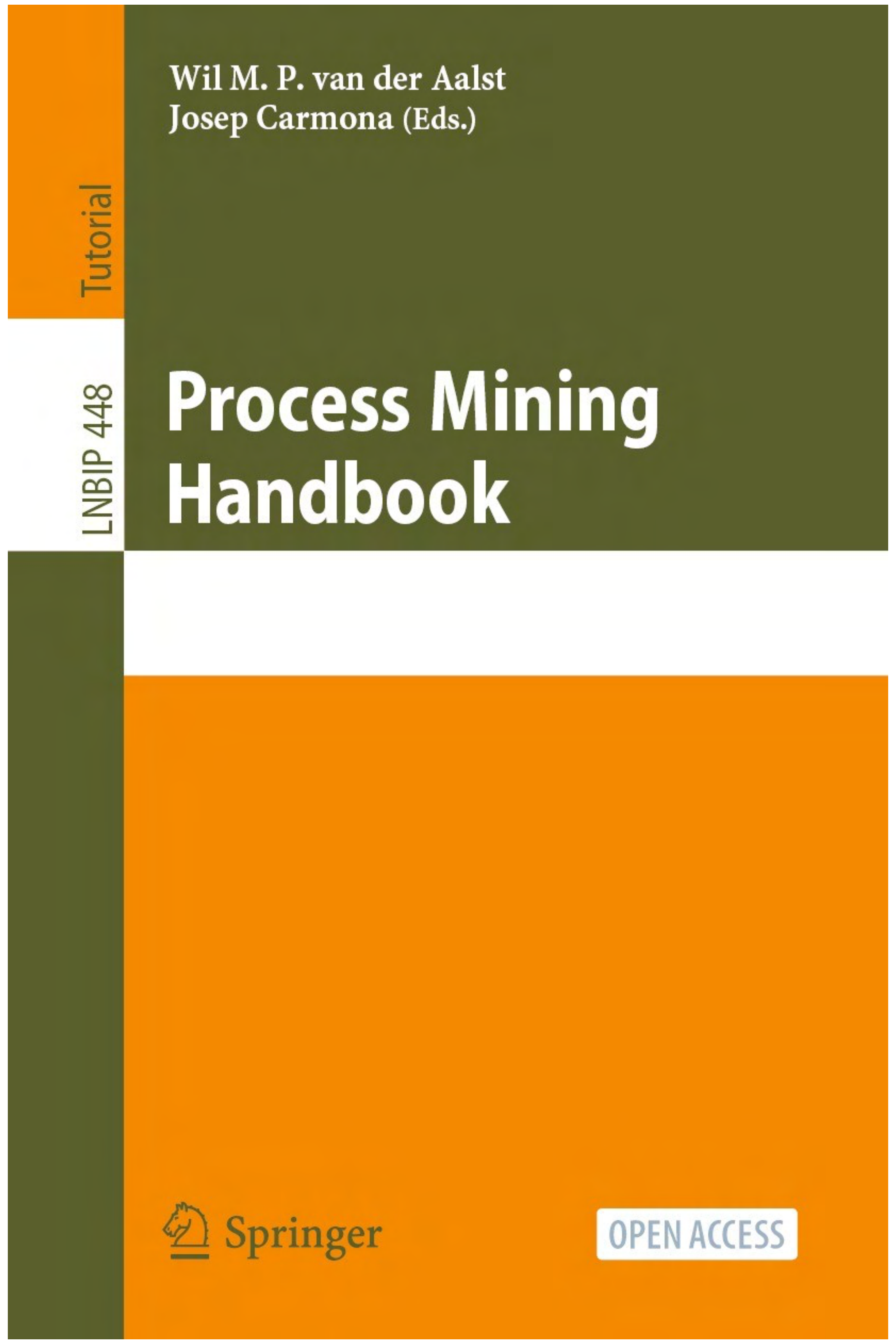
Wil van der Aalst

Process Mining

Data Science in Action

Second Edition

Springer



Tutorial

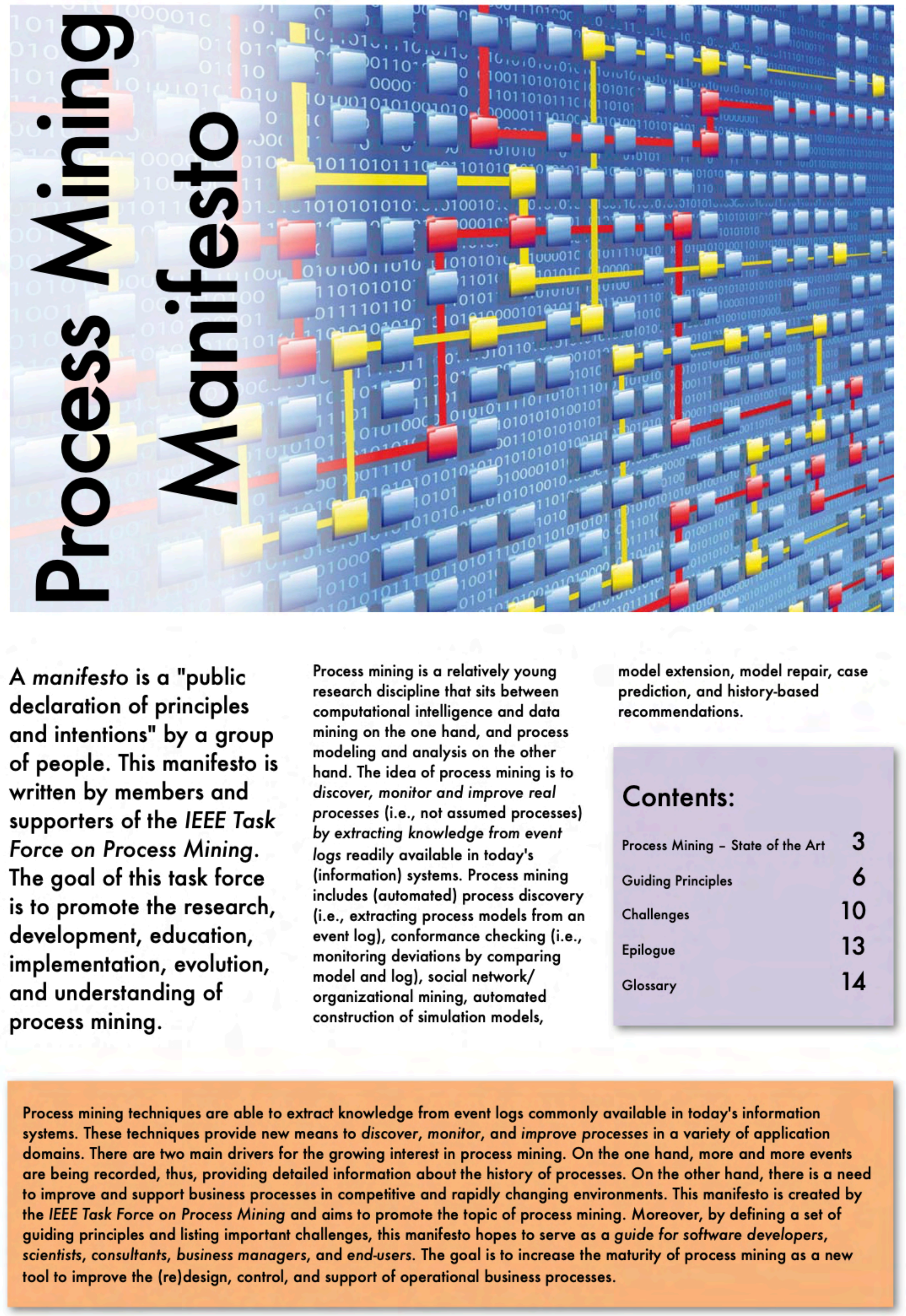
LNBIP 448

Wil M. P. van der Aalst
Josep Carmona (Eds.)

Process Mining Handbook

Springer

OPEN ACCESS



Process Mining Manifesto

A *manifesto* is a "public declaration of principles and intentions" by a group of people. This manifesto is written by members and supporters of the *IEEE Task Force on Process Mining*. The goal of this task force is to promote the research, development, education, implementation, evolution, and understanding of process mining.

Process mining is a relatively young research discipline that sits between computational intelligence and data mining on the one hand, and process modeling and analysis on the other hand. The idea of process mining is to *discover, monitor and improve real processes* (i.e., not assumed processes) *by extracting knowledge from event logs* readily available in today's (information) systems. Process mining includes (automated) process discovery (i.e., extracting process models from an event log), conformance checking (i.e., monitoring deviations by comparing model and log), social network/organizational mining, automated construction of simulation models, model extension, model repair, case prediction, and history-based recommendations.

Contents:

Process mining techniques are able to extract knowledge from event logs commonly available in today's information systems. These techniques provide new means to *discover*, *monitor*, and *improve processes* in a variety of application domains. There are two main drivers for the growing interest in process mining. On the one hand, more and more events are being recorded, thus, providing detailed information about the history of processes. On the other hand, there is a need to improve and support business processes in competitive and rapidly changing environments. This manifesto is created by the *IEEE Task Force on Process Mining* and aims to promote the topic of process mining. Moreover, by defining a set of guiding principles and listing important challenges, this manifesto hopes to serve as a *guide for software developers, scientists, consultants, business managers*, and *end-users*. The goal is to increase the maturity of process mining as a new tool to improve the (re)design, control, and support of operational business processes.

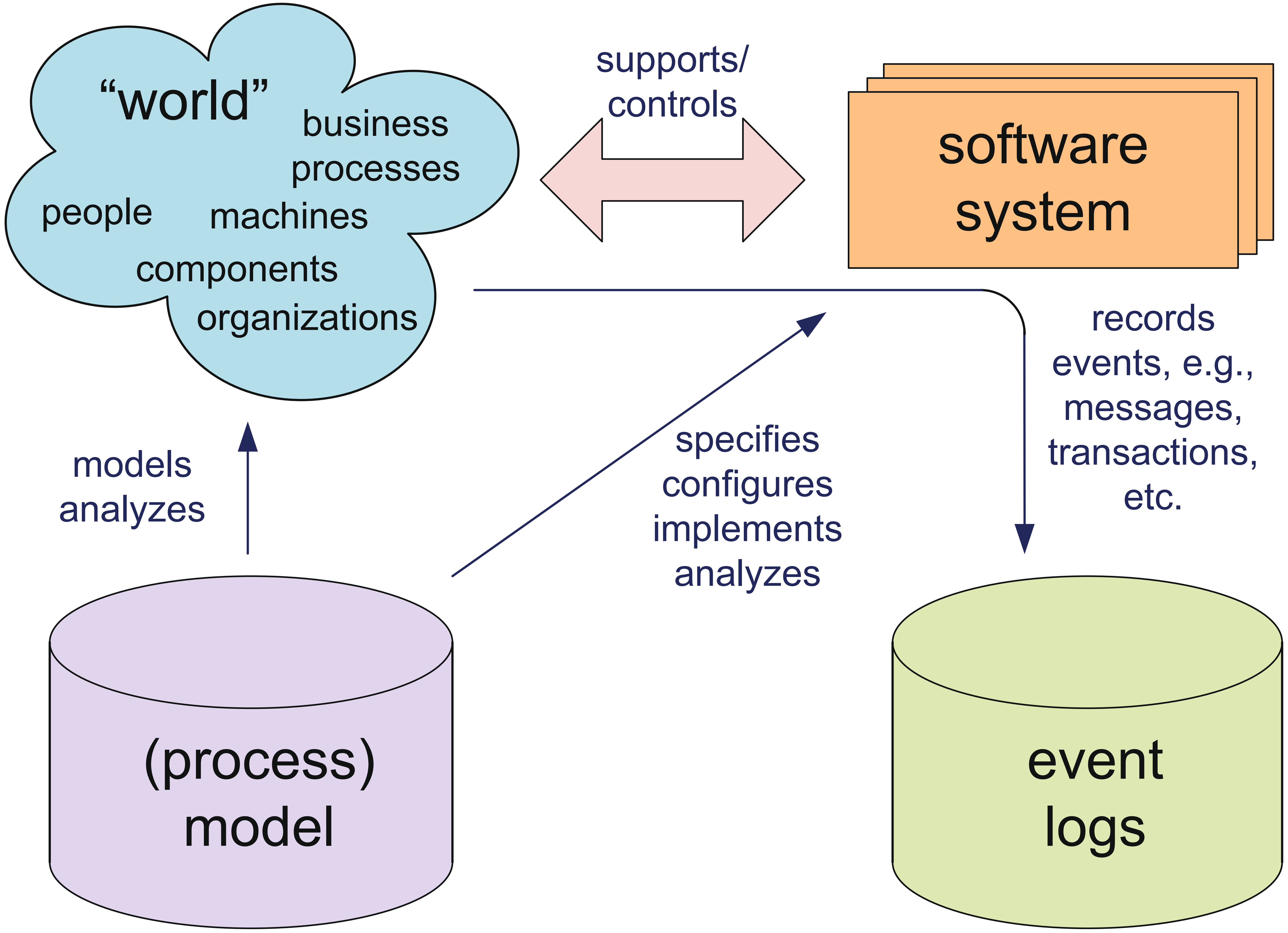
TFPM

IEEE TASK FORCE ON PROCESS MINING

https://www.tf-pm.org

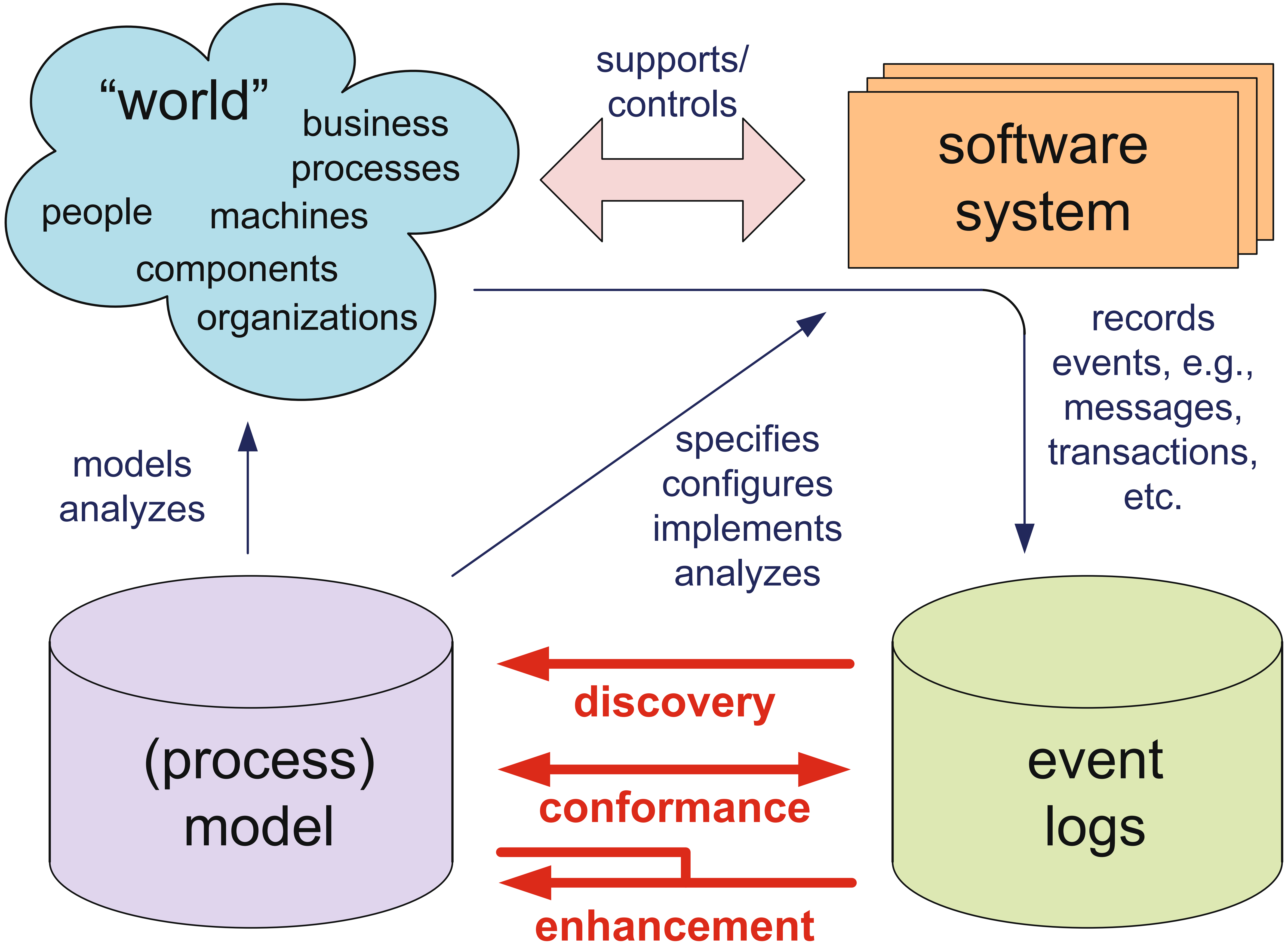
The process mining framework

Original picture by Wil van der Aalst



The process mining framework

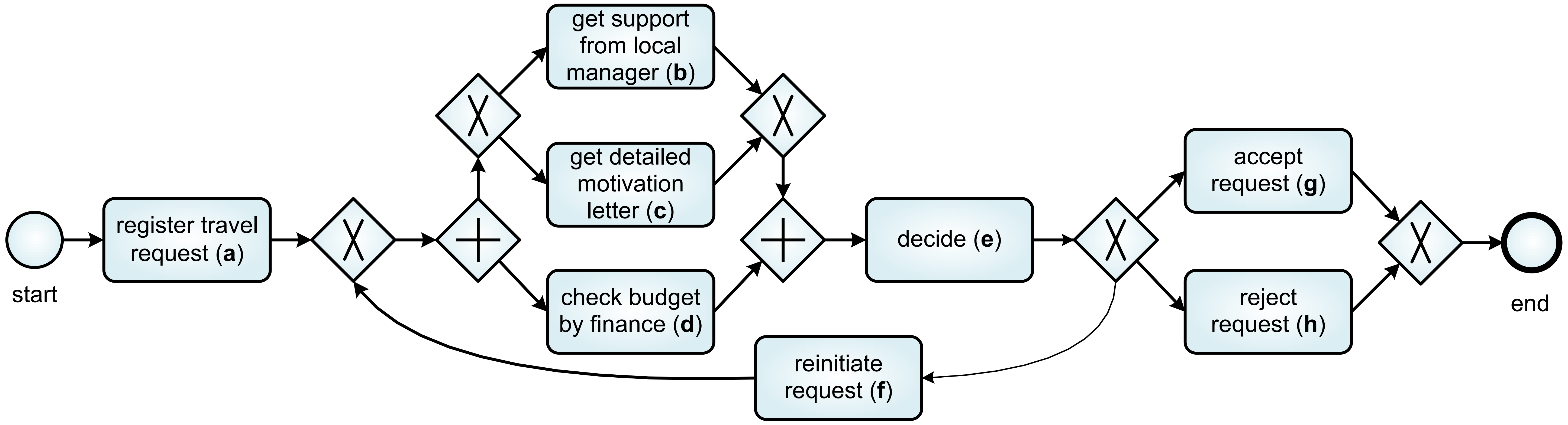
Original picture by Wil van der Aalst



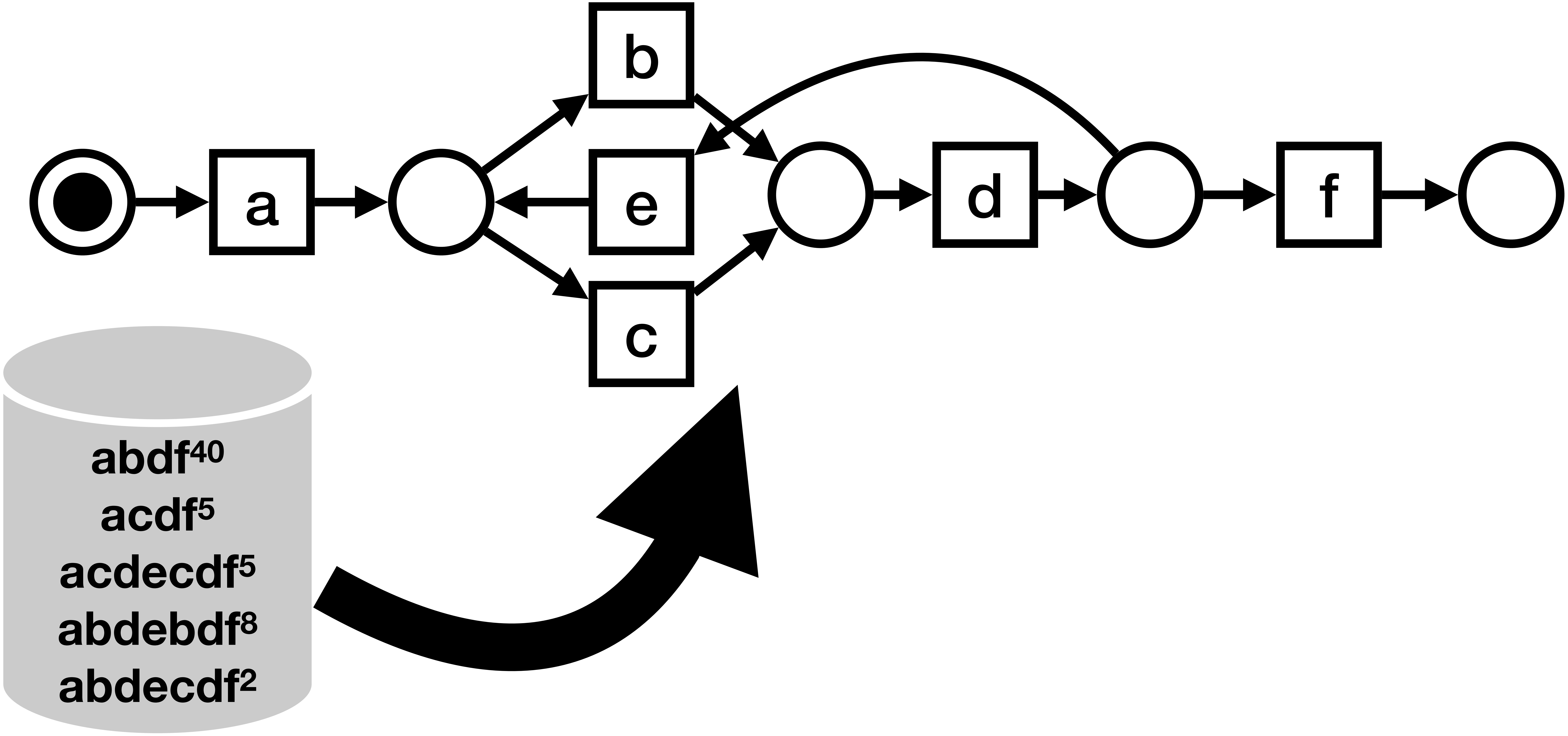
Play in: discovery

Case	Activity	Timestamp	Resource
432	register travel request (a)	18-3-2014:9.15	John
432	get support from local manager (b)	18-3-2014:9.25	Mary
432	check budget by finance (d)	19-3-2014:8.55	John
432	decide (e)	19-3-2014:9.36	Sue
432	accept request (g)	19-3-2014:9.48	Mary

...


start
register travel request (a)
get support from local manager (b)
get detailed motivation letter (c)
check budget by finance (d)
decide (e)
reinitiate request (f)
accept request (g)
reject request (h)
end

Replay: enhancement



$abdf^{40}$

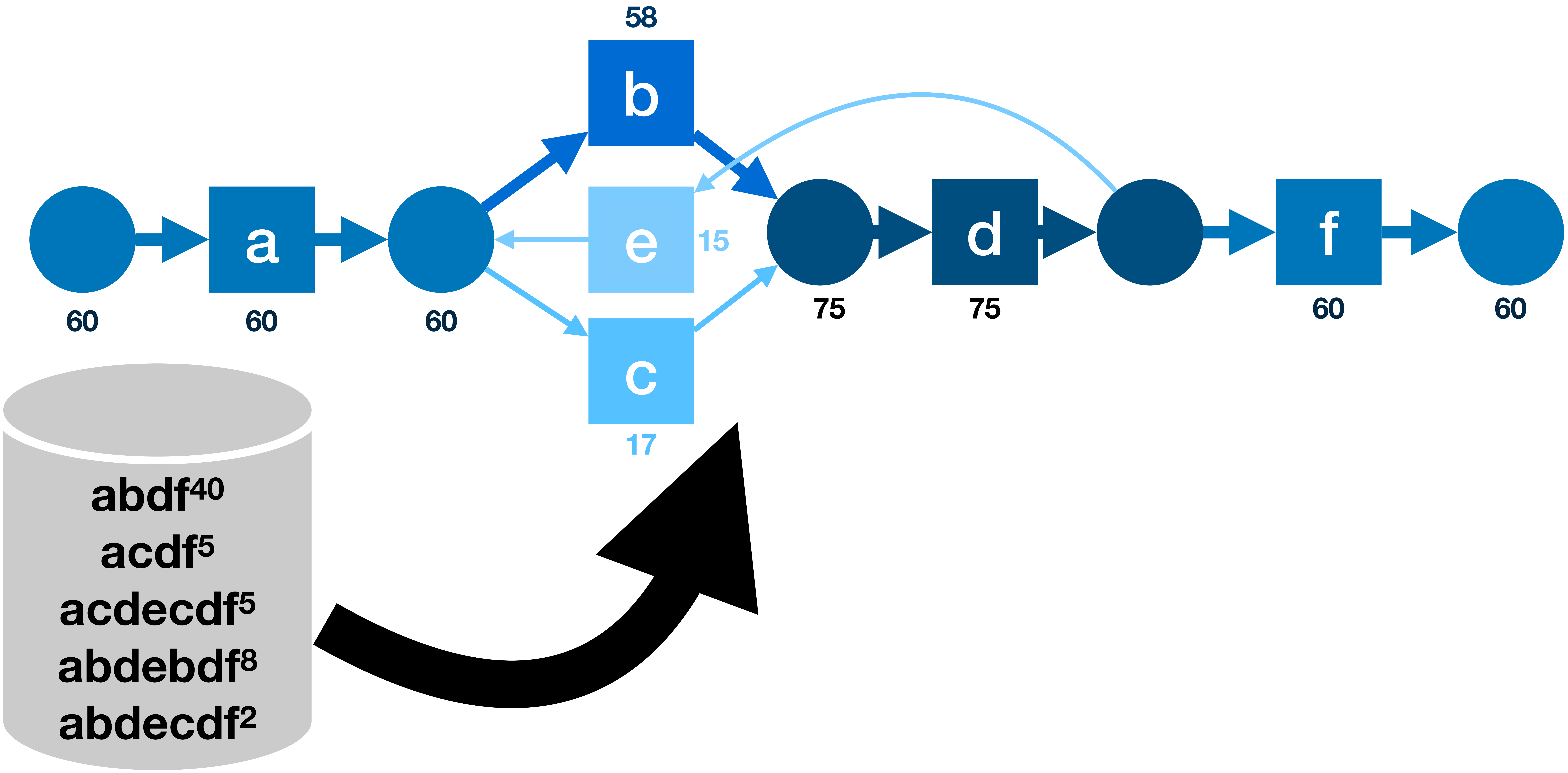
$acdf^{5}$

$acdecdf^{5}$

$abdebdf^{8}$

$abdecdf^{2}$

Replay: enhancement



abdf^{40}

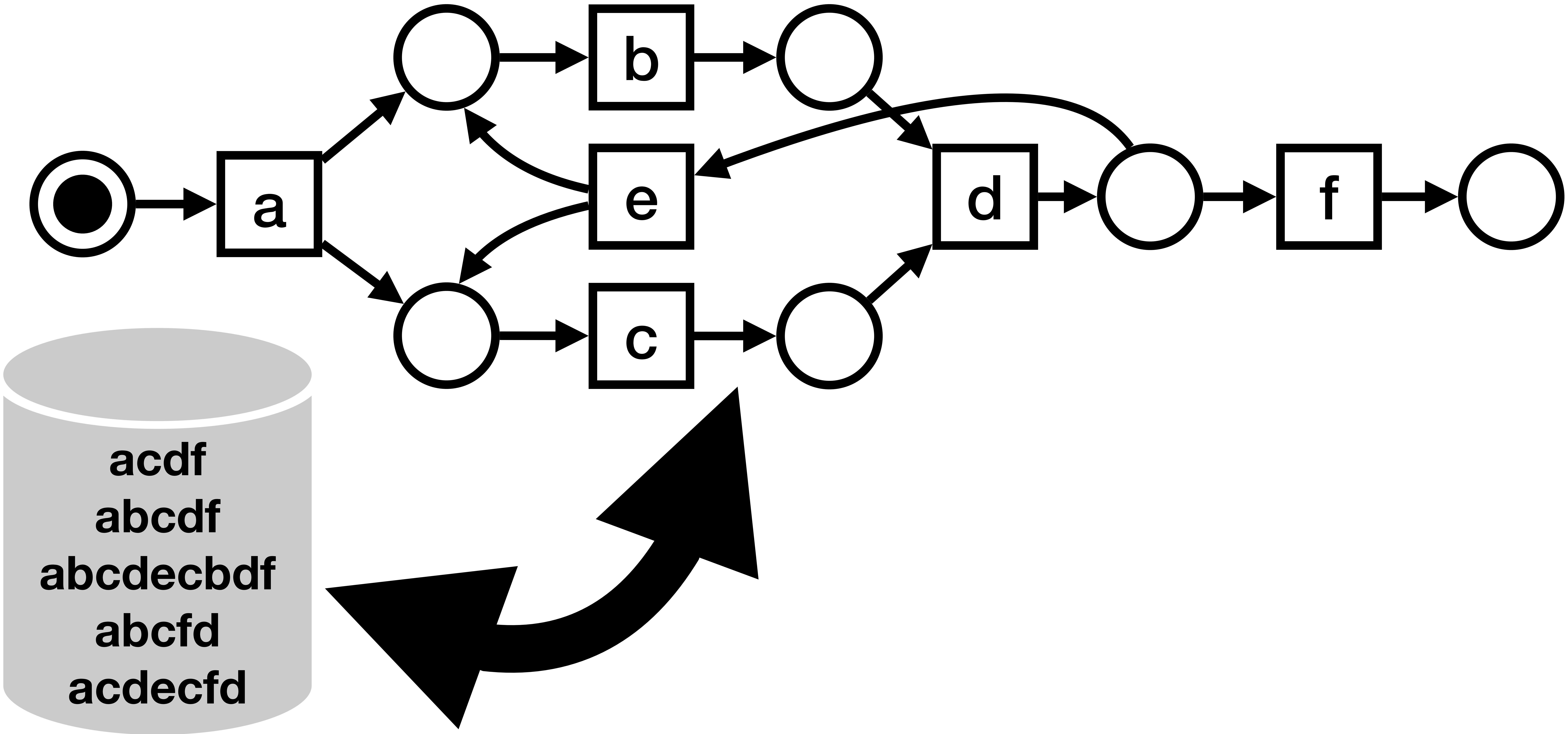
acdf^{5}

acdecdf^{5}

abdebdf^{8}

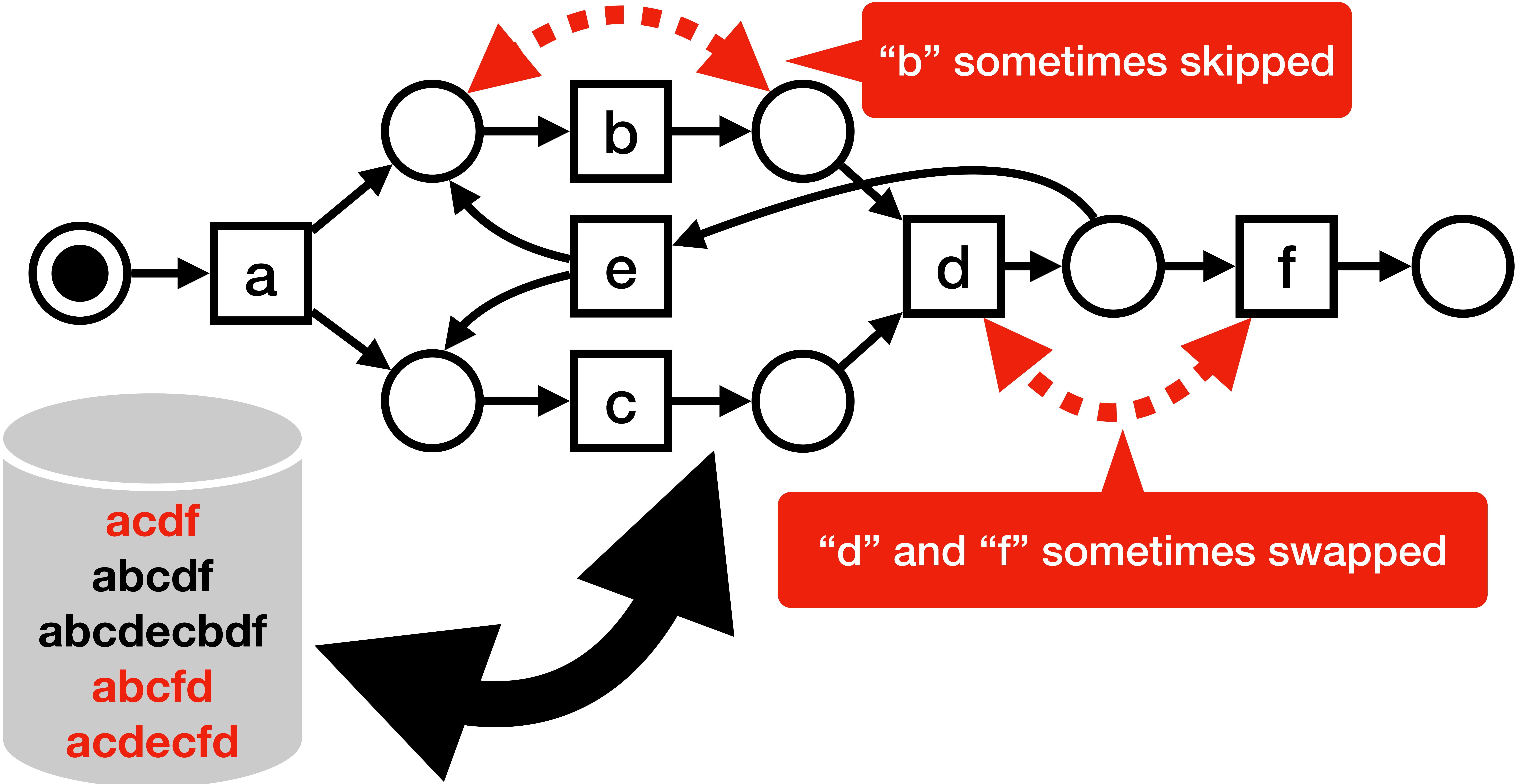
abdecdf^{2}

Replay: conformance checking



acdf
abcdf
abcdecbdf
abcfd
acdecfd

Replay: conformance checking

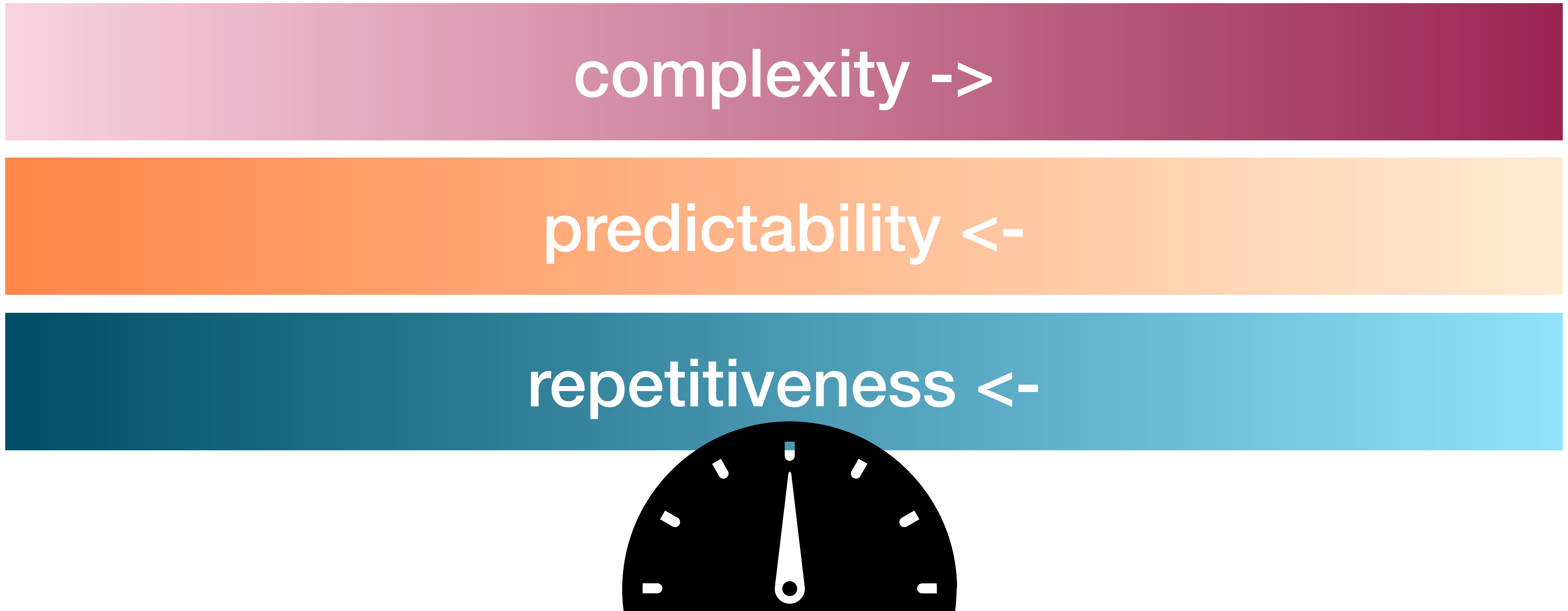


“b” sometimes skipped

“d” and “f” sometimes swapped

Are all processes the same ?

On control and flexibility



complexity ->

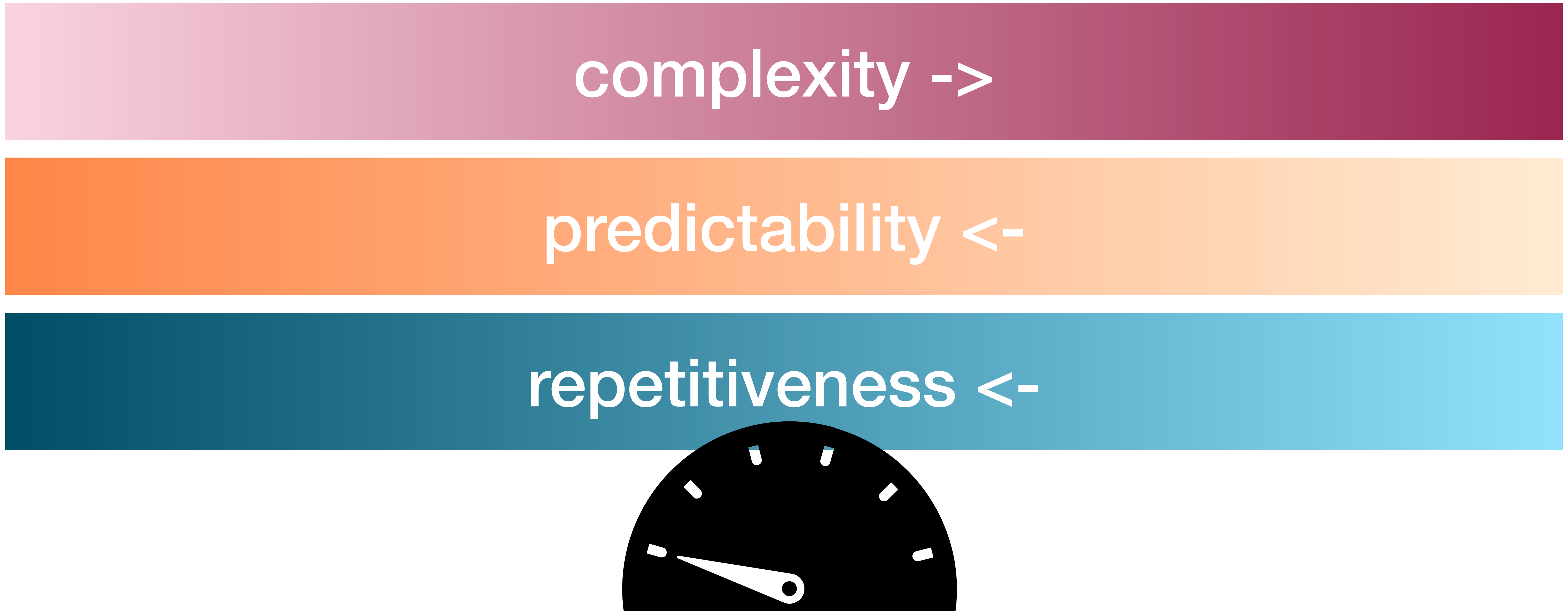
predictability <-

repetitiveness <-

On control and flexibility

Control

degree to which a central orchestrator decides how to execute the process



On control and flexibility

Control

degree to which a central orchestrator decides how to execute the process

Flexibility

degree to which process stakeholders locally decide how to execute the process

complexity ->

predictability <-

repetitiveness <-

On control and flexibility

Control

degree to which a central orchestrator decides how to execute the process

Flexibility

degree to which process stakeholders locally decide how to execute the process

complexity ->

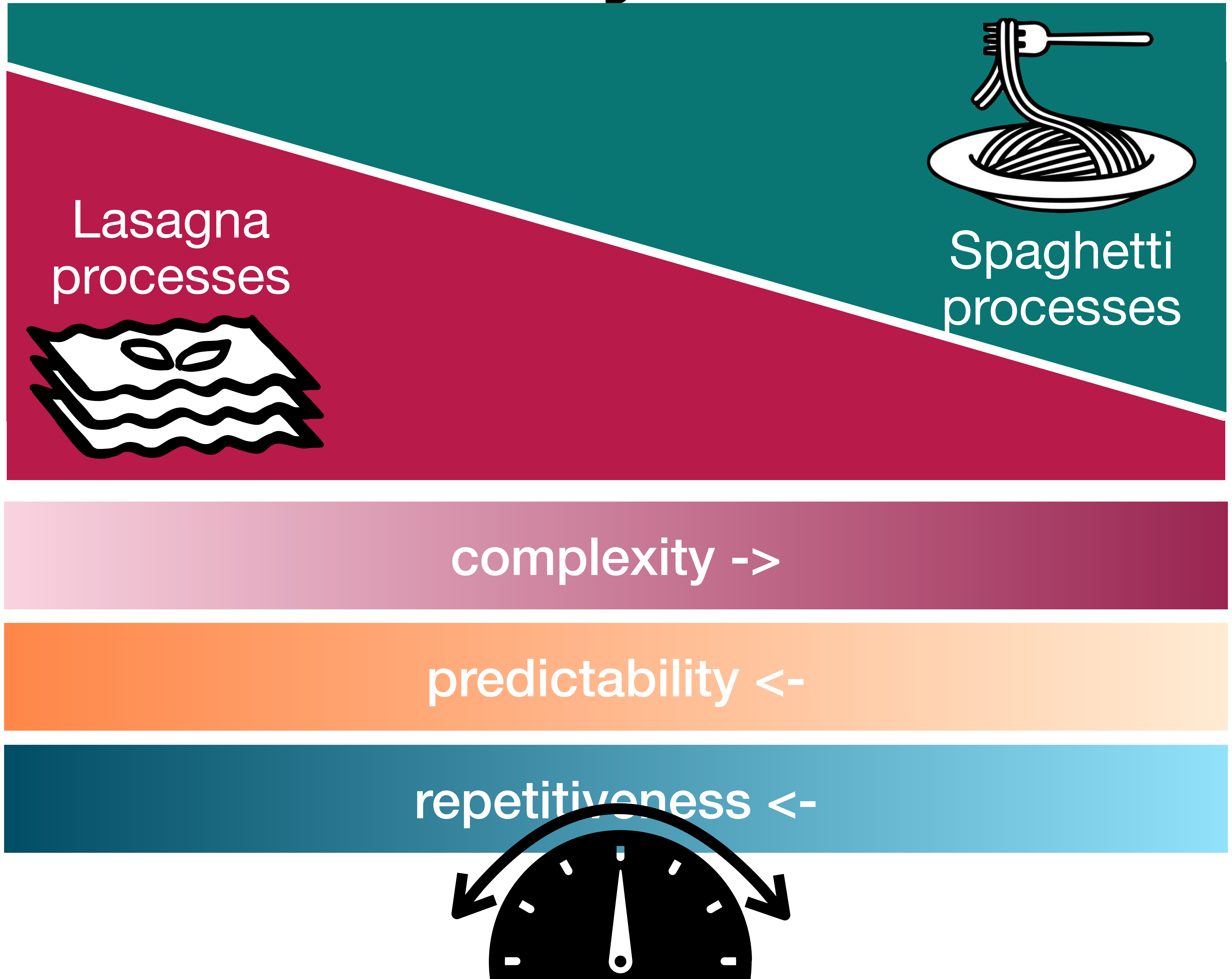
predictability <-

repetitiveness <-

Which Italian food do you like best?

Control

degree to which a central orchestrator decides how to execute the process



Lasagna processes

Spaghetti processes

complexity ->

predictability <-

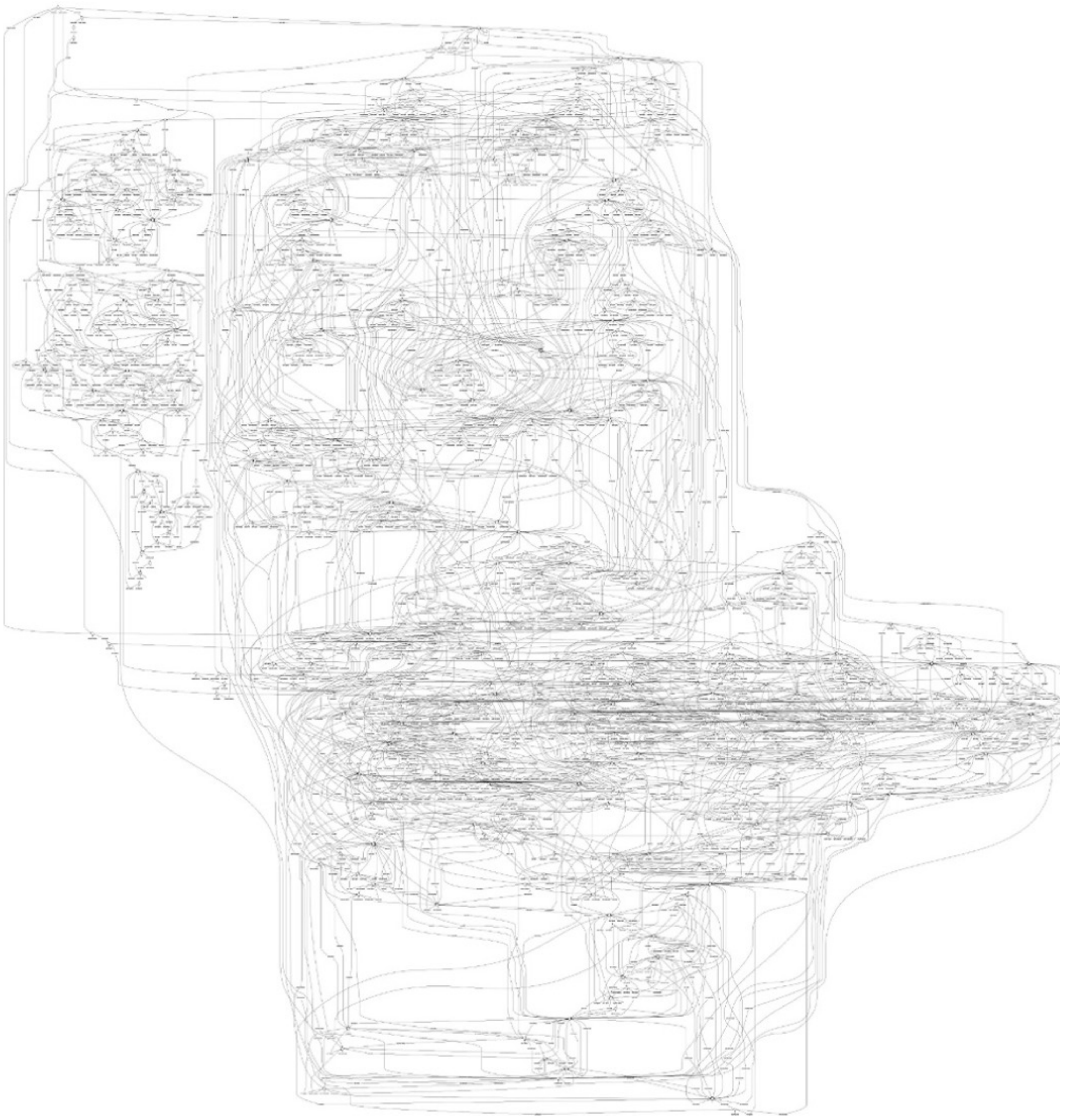
repetitiveness <-

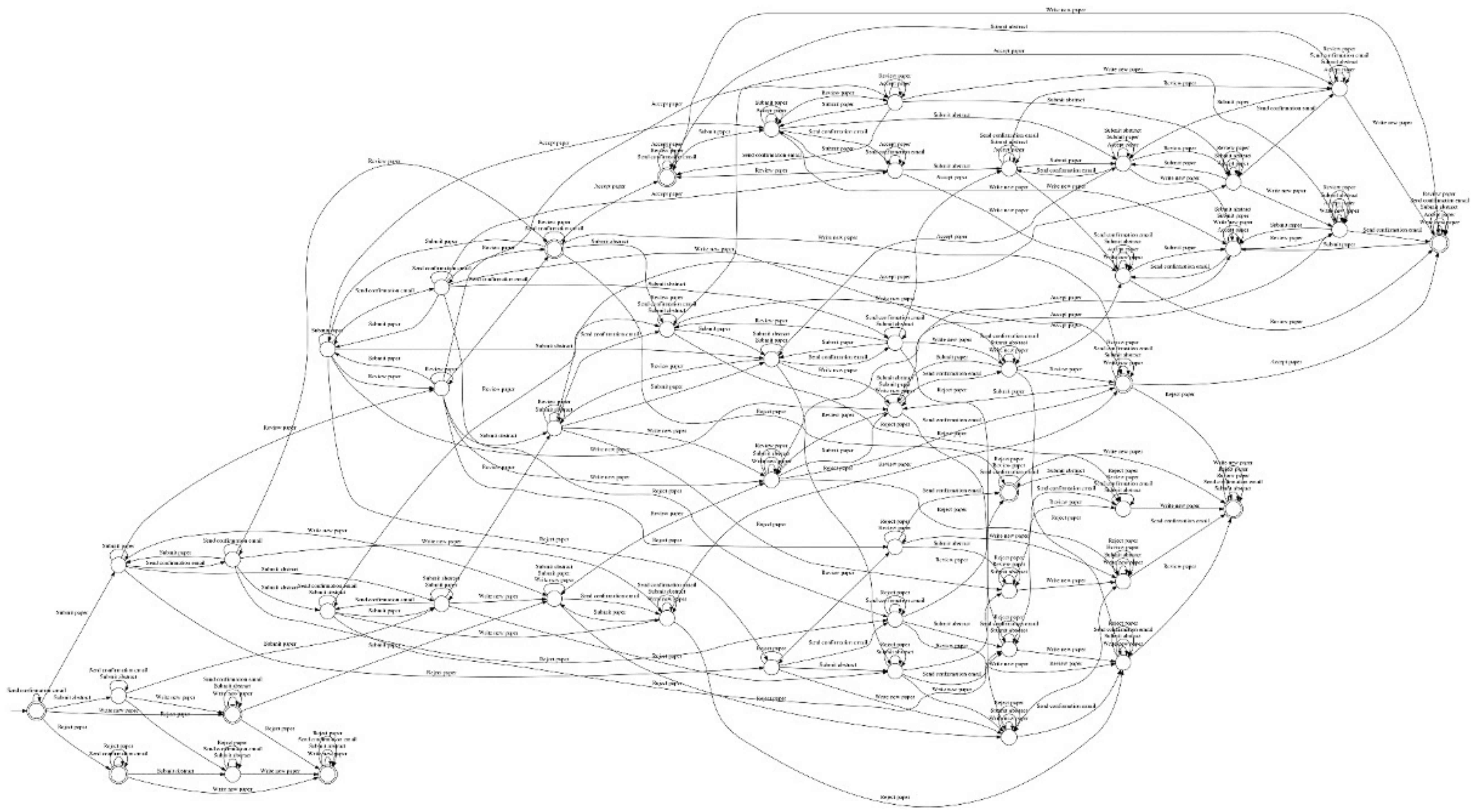
Flexibility

degree to which process stakeholders locally decide how to execute the process

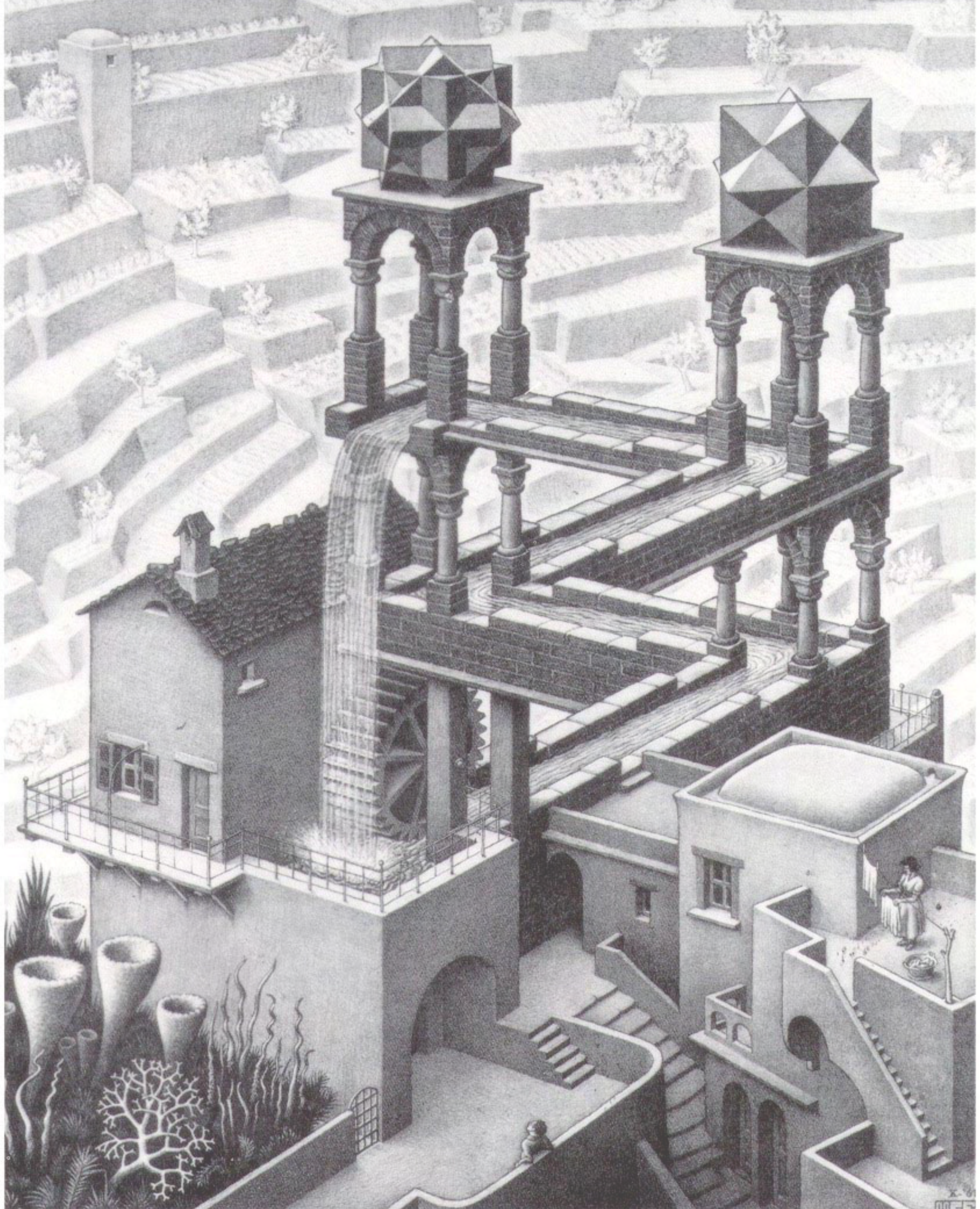
Reality is often more flexible than it seems…

Reality is often more flexible than it seems…

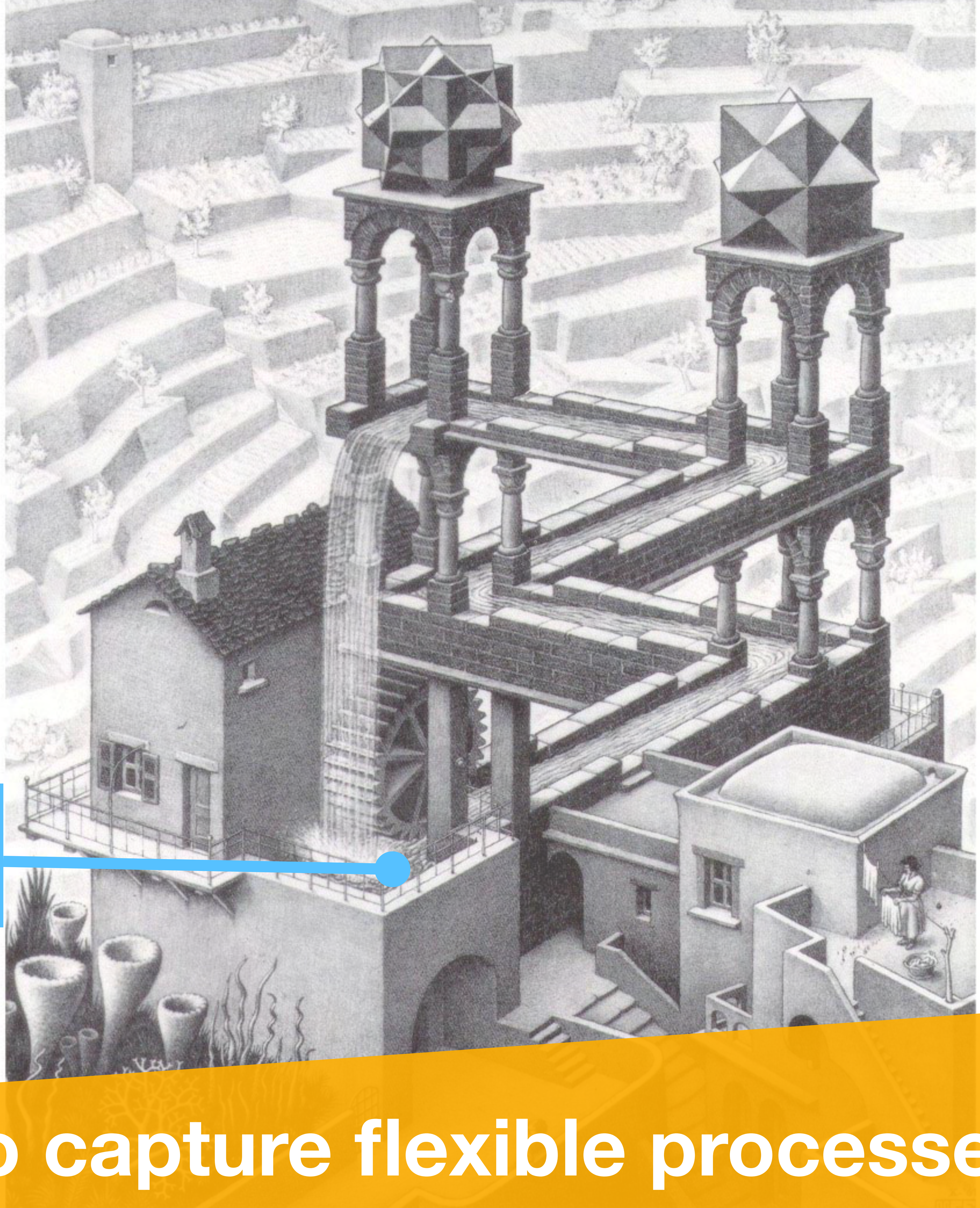




Outline



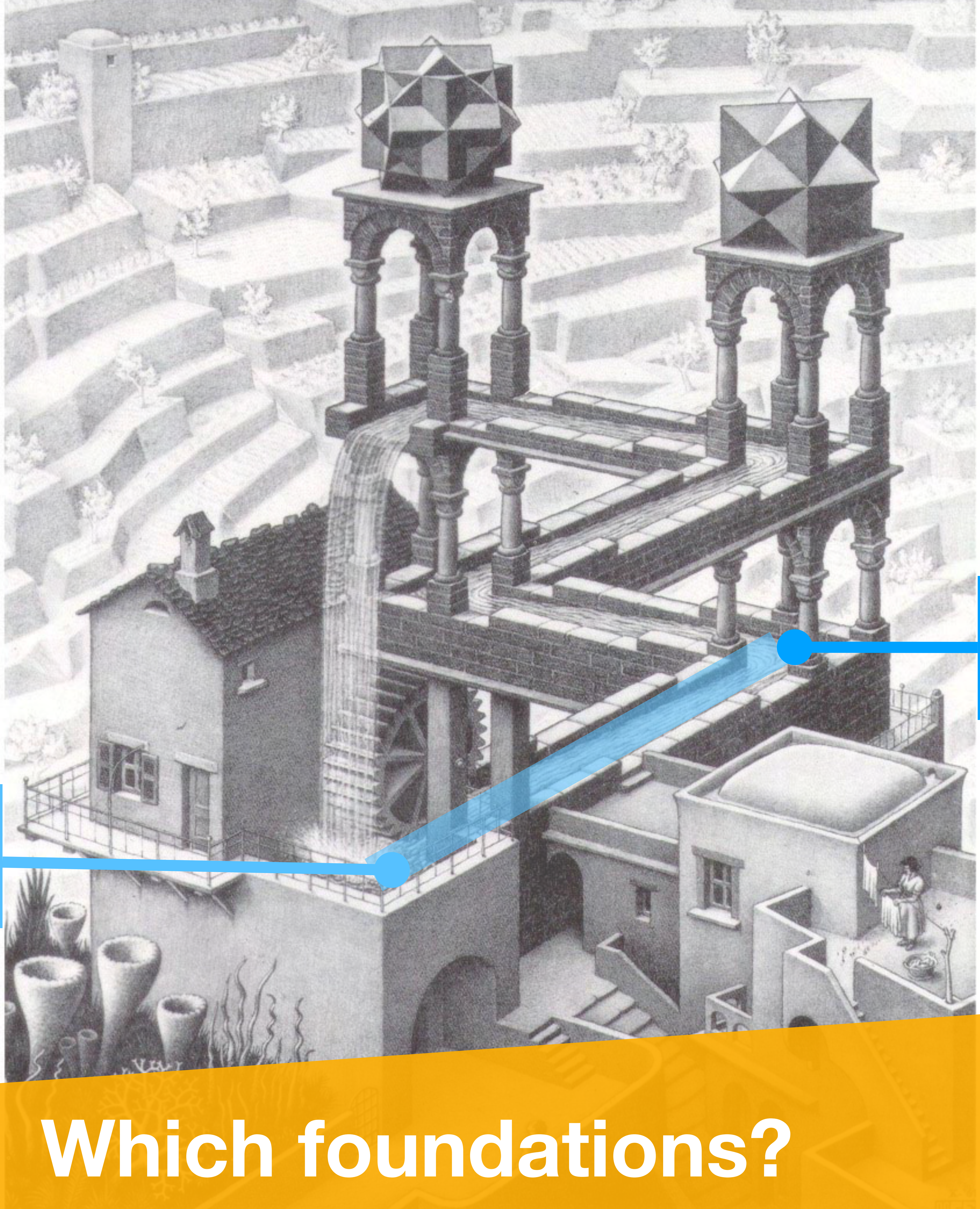
Outline



The DECLARE declarative approach

How to capture flexible processes?

Outline

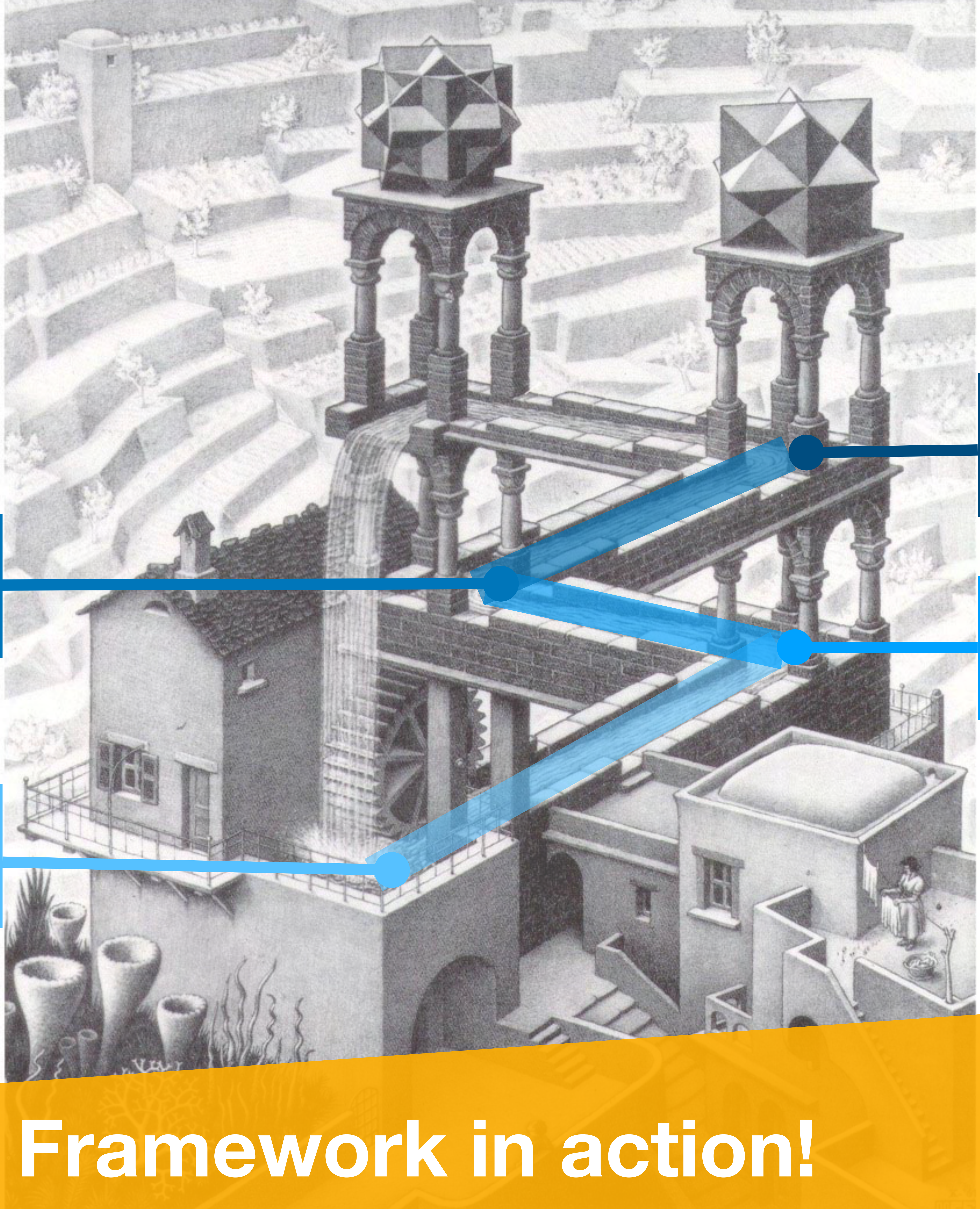


LTLf and automata to the rescue

The DECLARE declarative approach

Which foundations?

Outline



Declarative process discovery

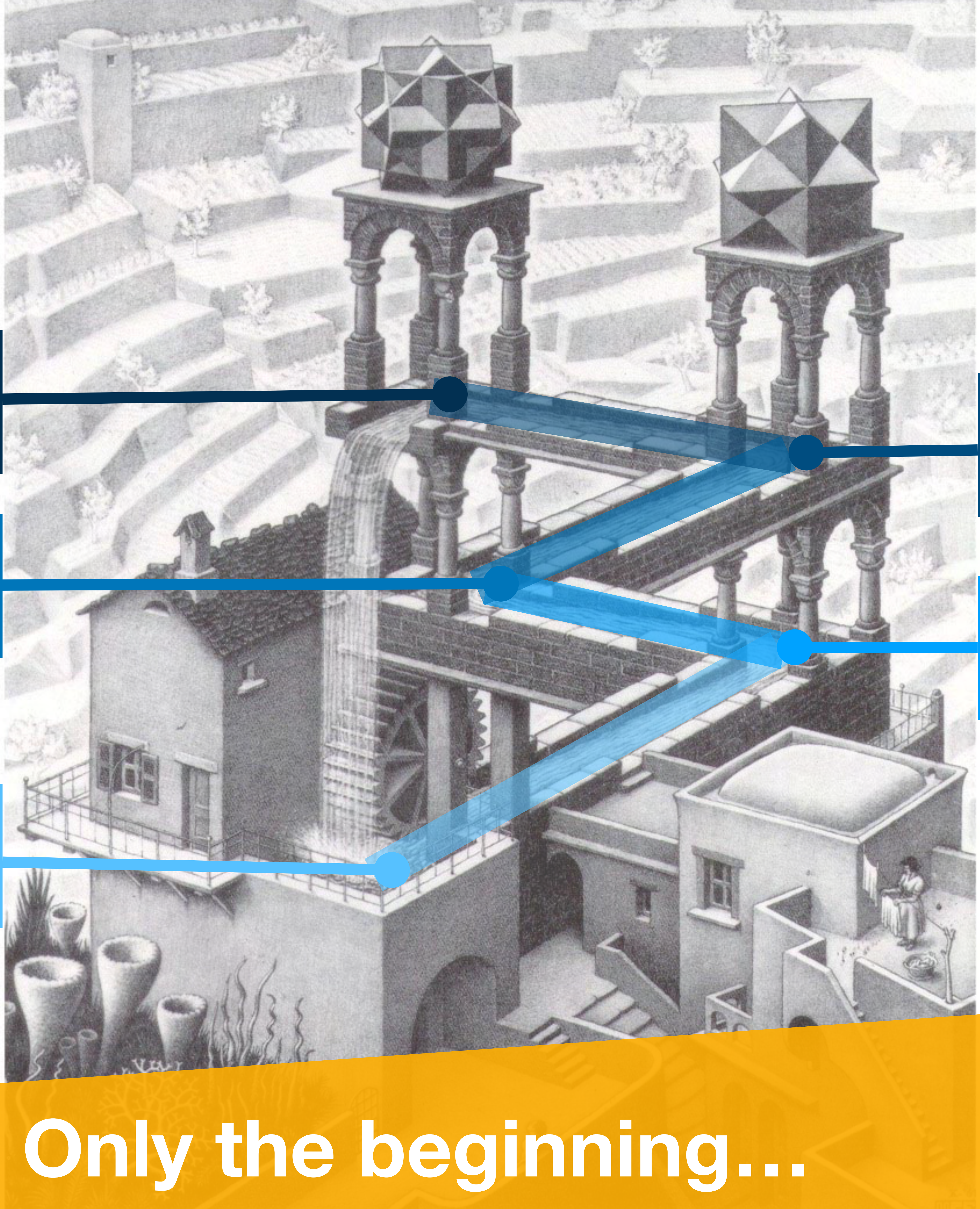
Enactment and monitoring

LTLf and automata to the rescue

The DECLARE declarative approach

Framework in action!

Outline



5 exciting research lines

Declarative process discovery

Enactment and monitoring

LTLf and automata to the rescue

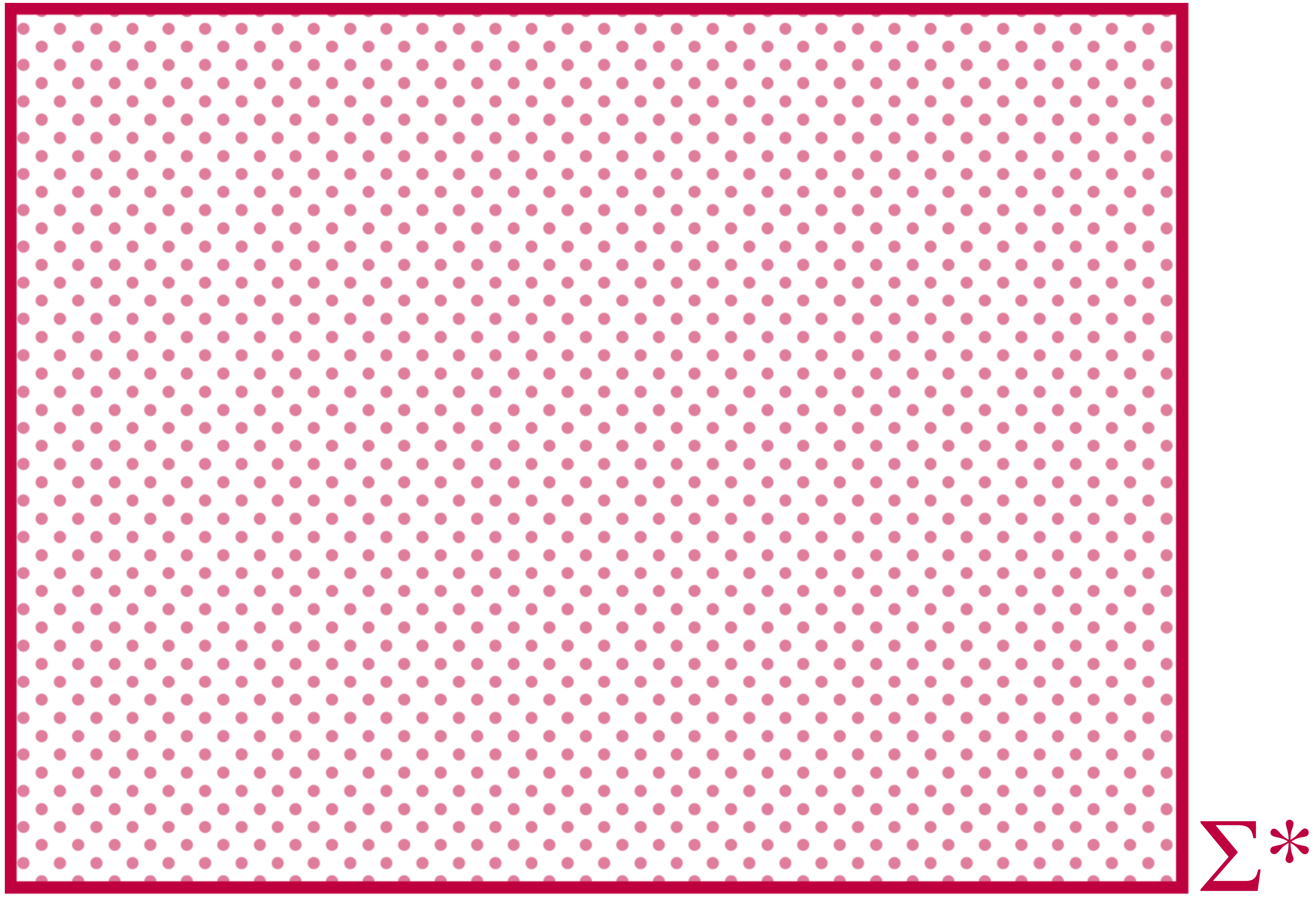
The DECLARE declarative approach

Only the beginning…

How to capture flexible processes?

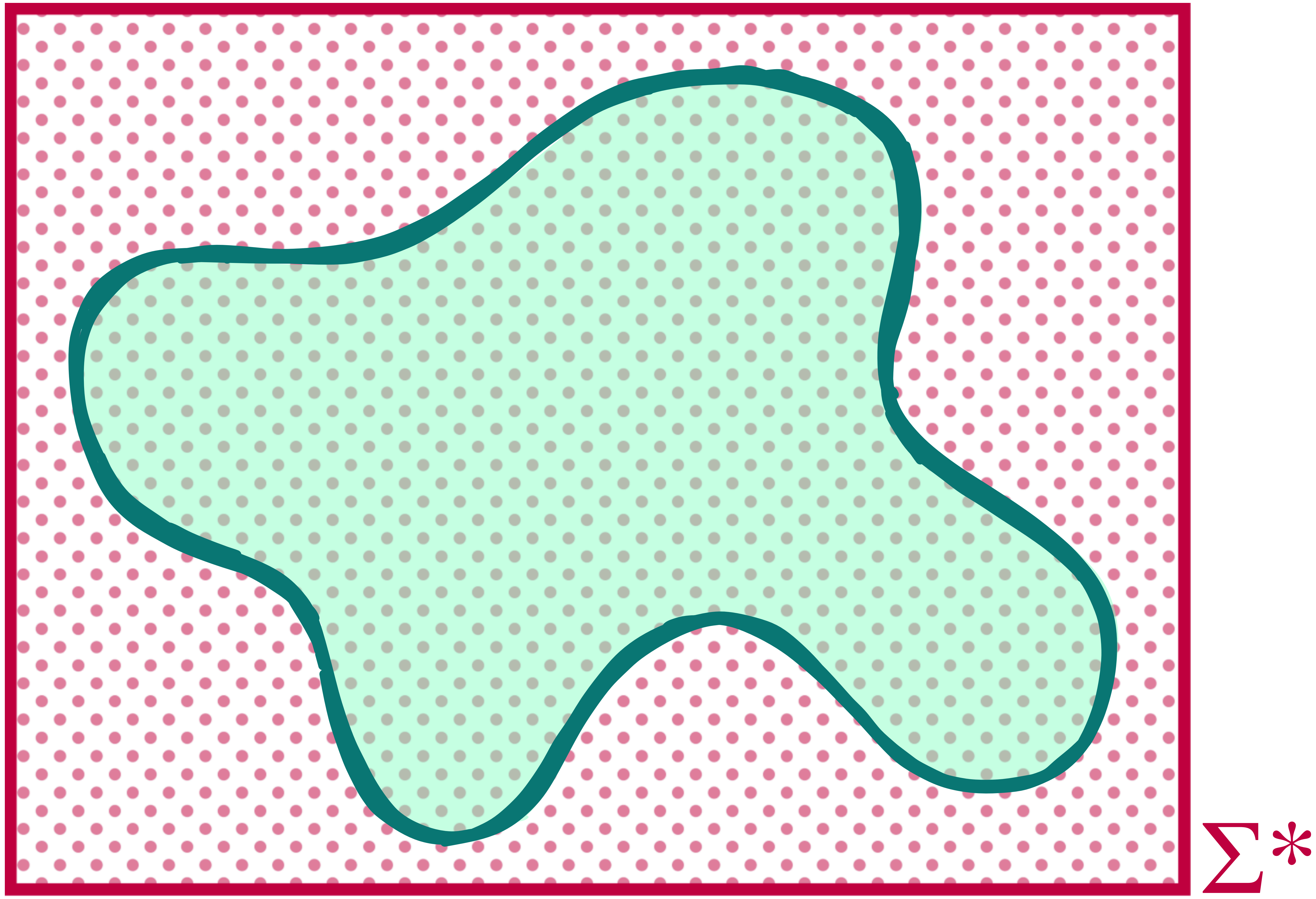
What is a process?

A possibly infinite set of finite traces

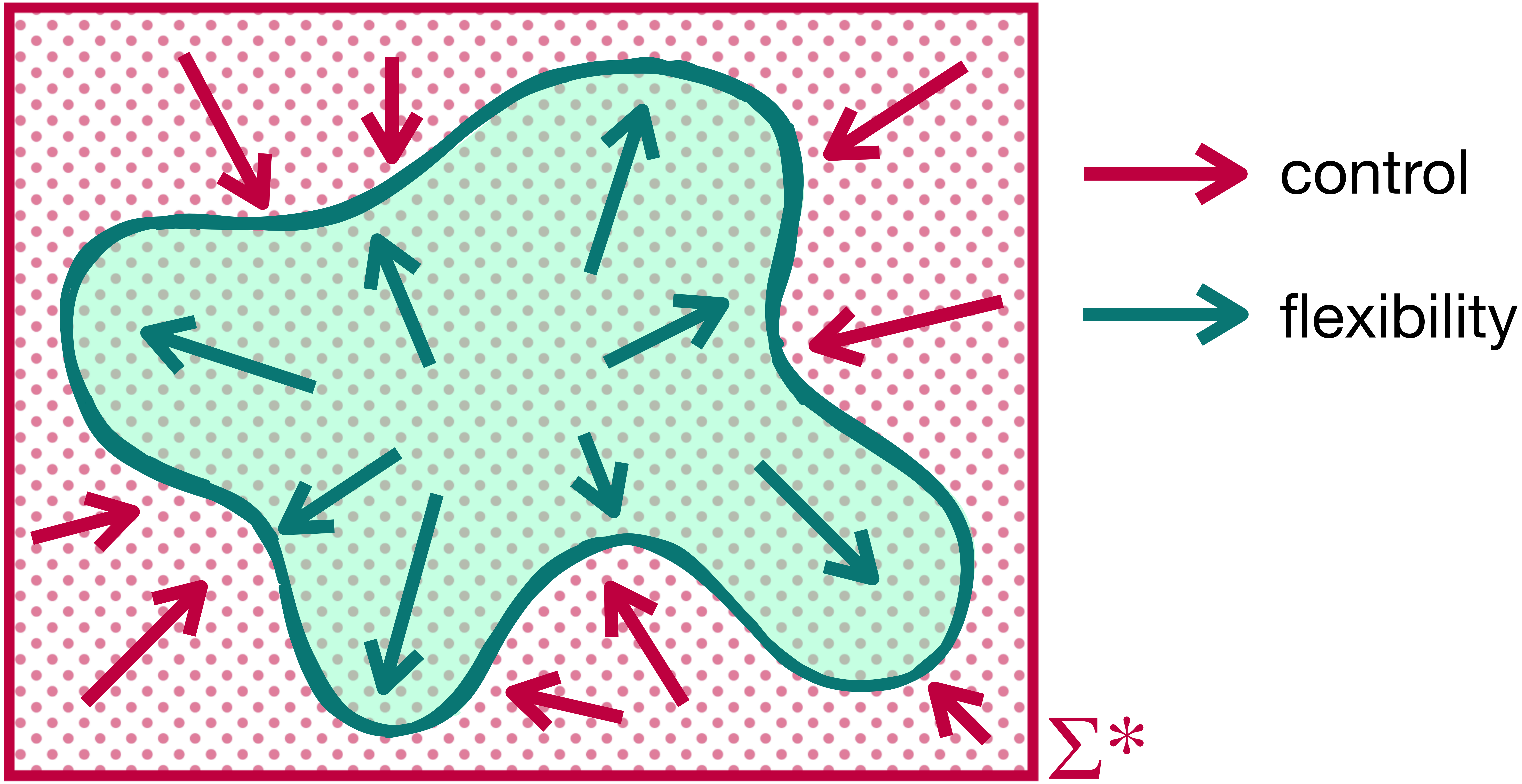


What is a process?

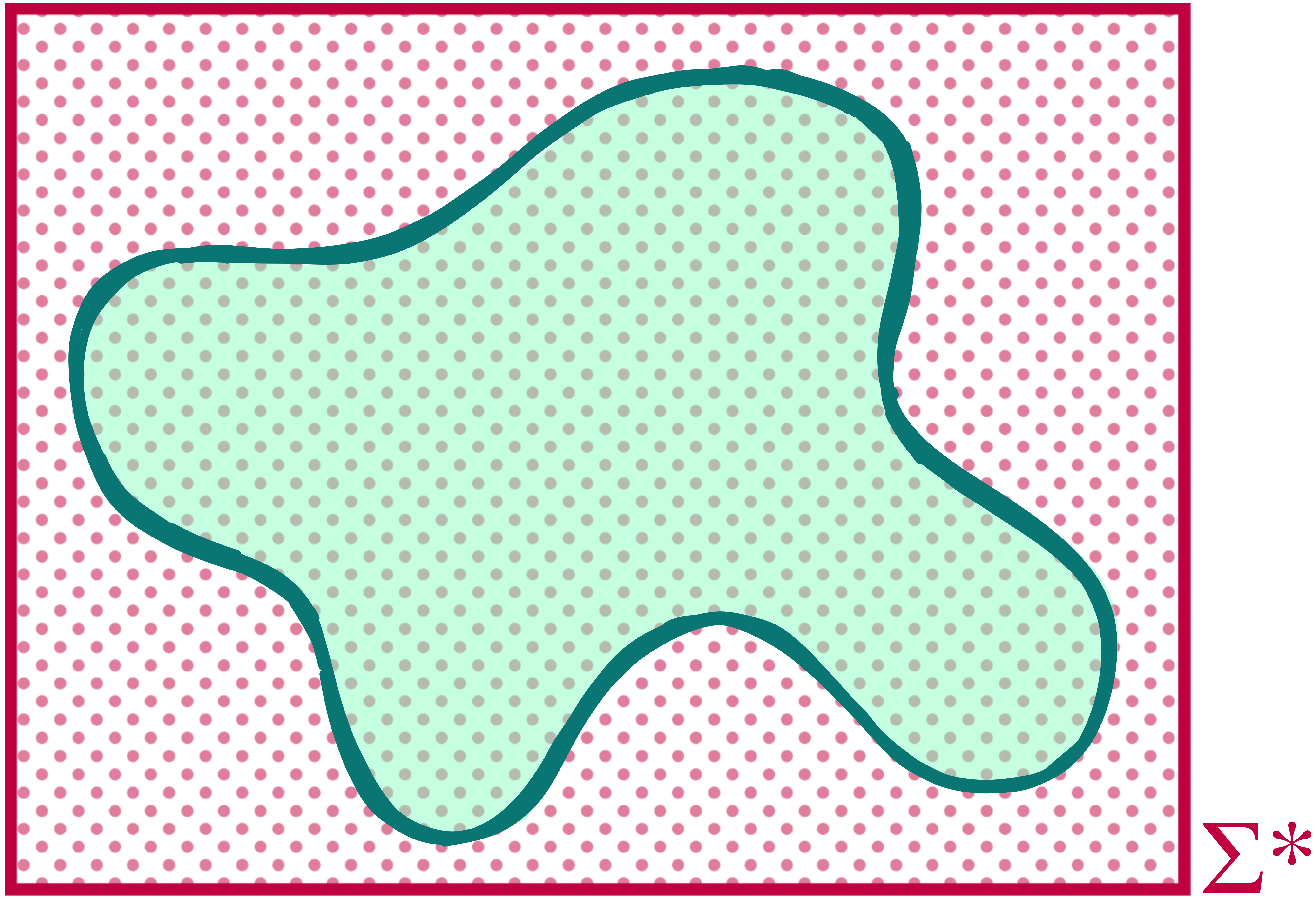
A possibly infinite set of finite traces



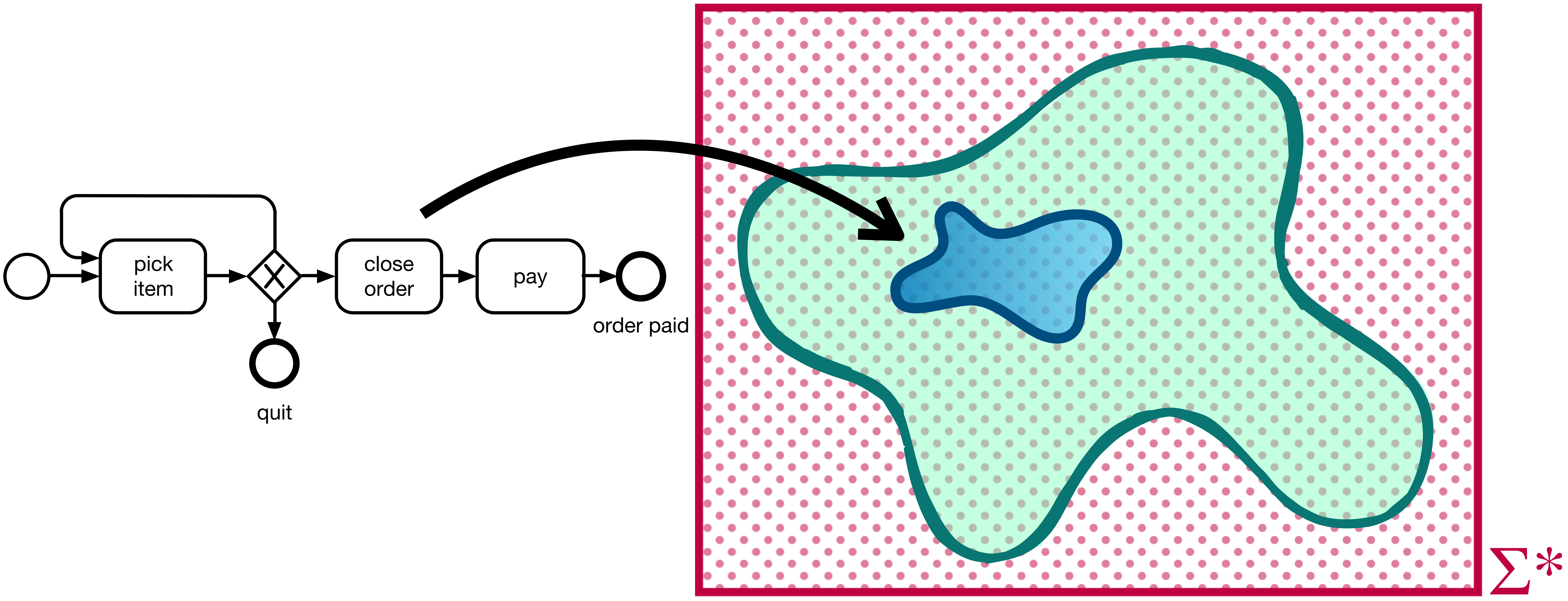
Flexibility and control as contrasting forces



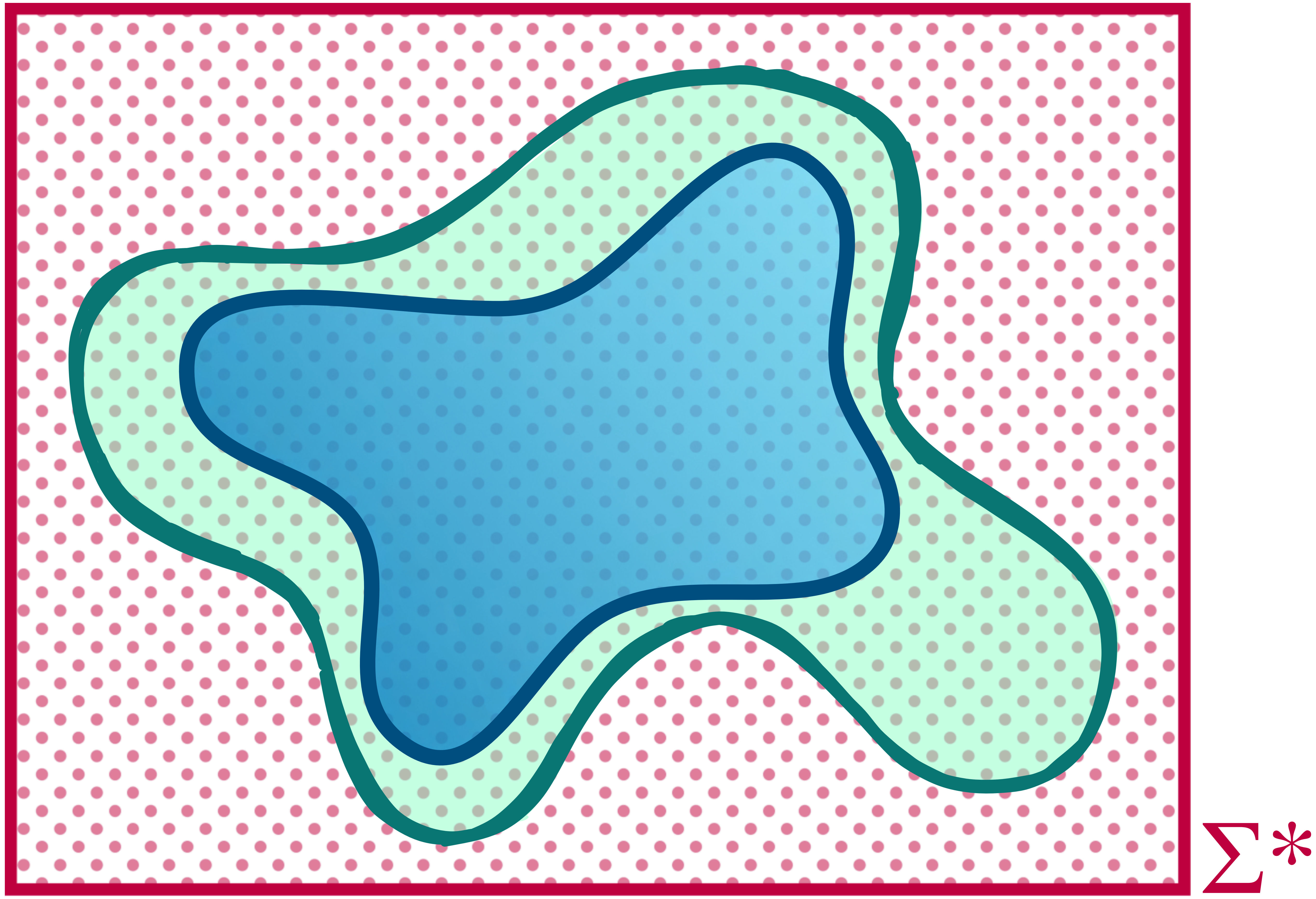
A process…



… and an imperative model of it



Generalisation



Generalisation

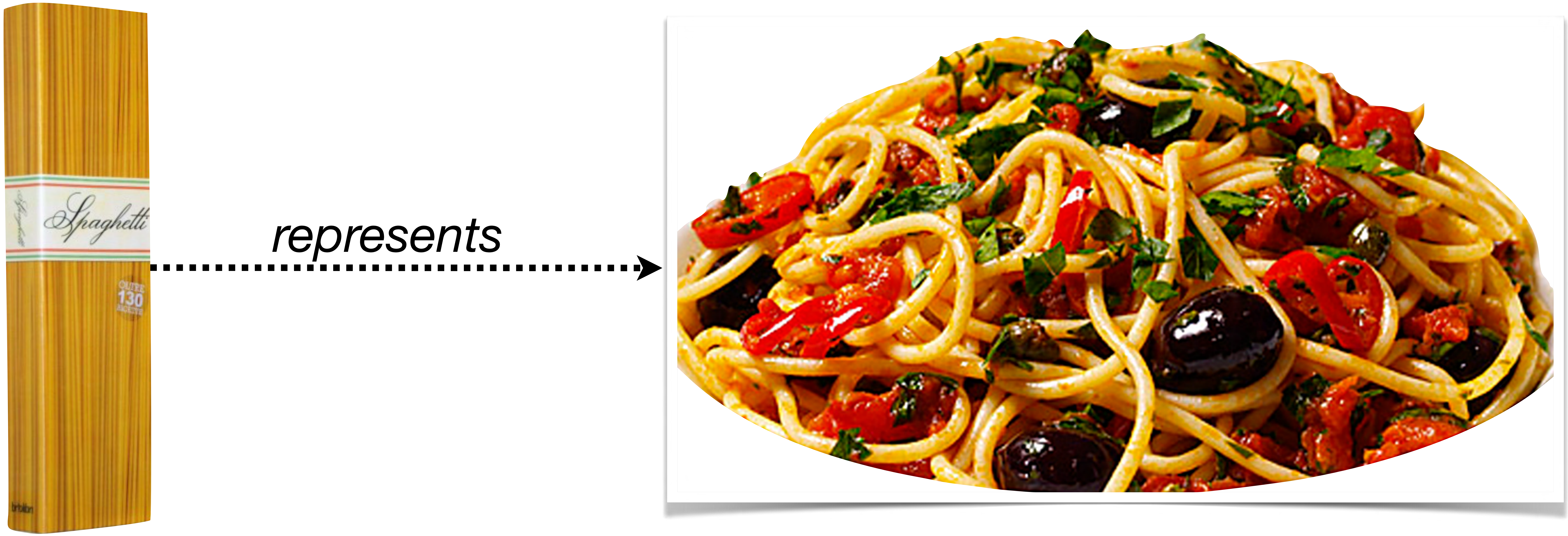
pick item
pay
close order
pick item
pay
Σ^*

The declarative approach

⚠

Simplicity cannot be obtained by sweeping complexity under the carpet

Our goal



represents

Compact specification

Reality

Our goal

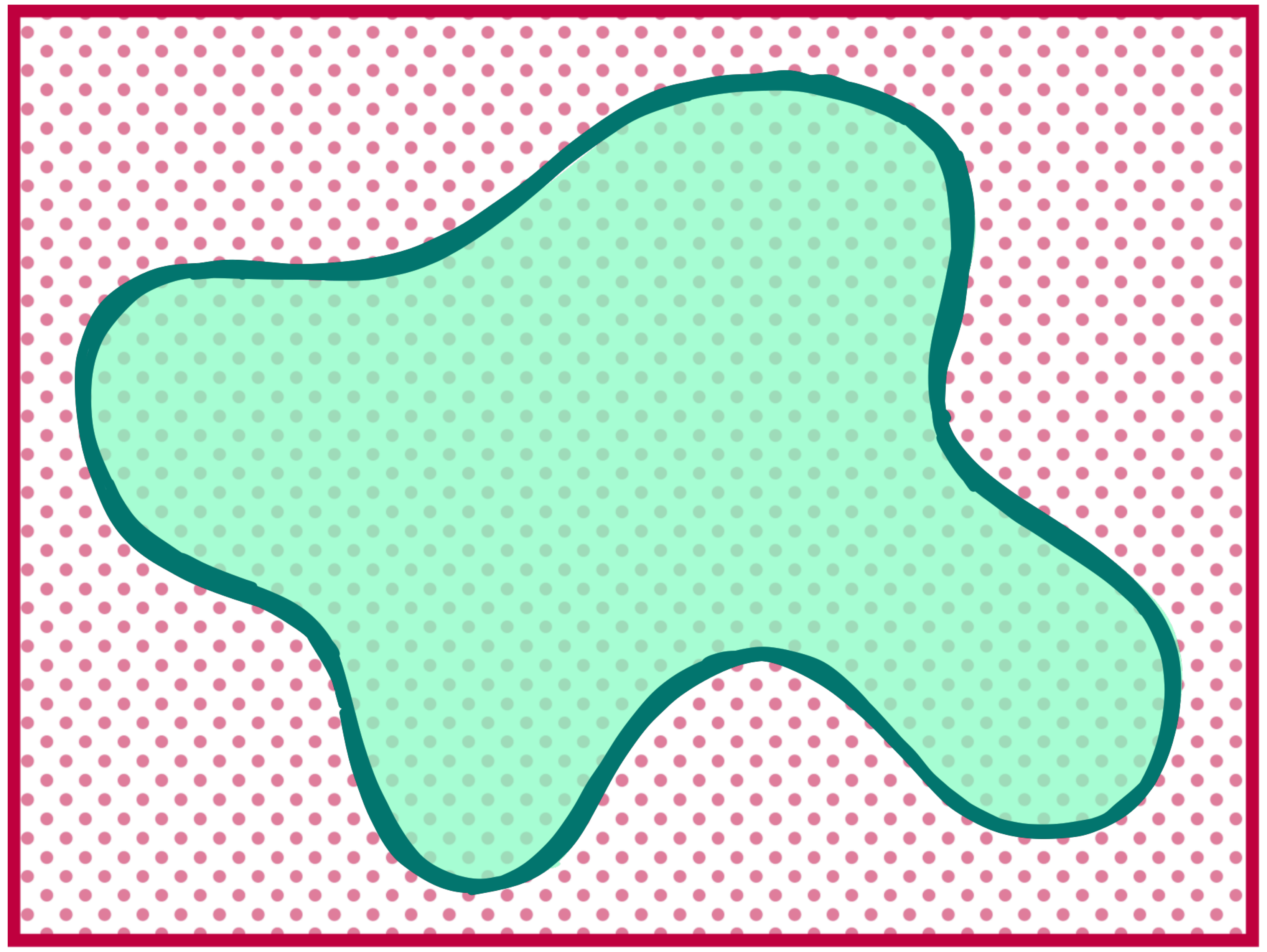
Compact specification

represents

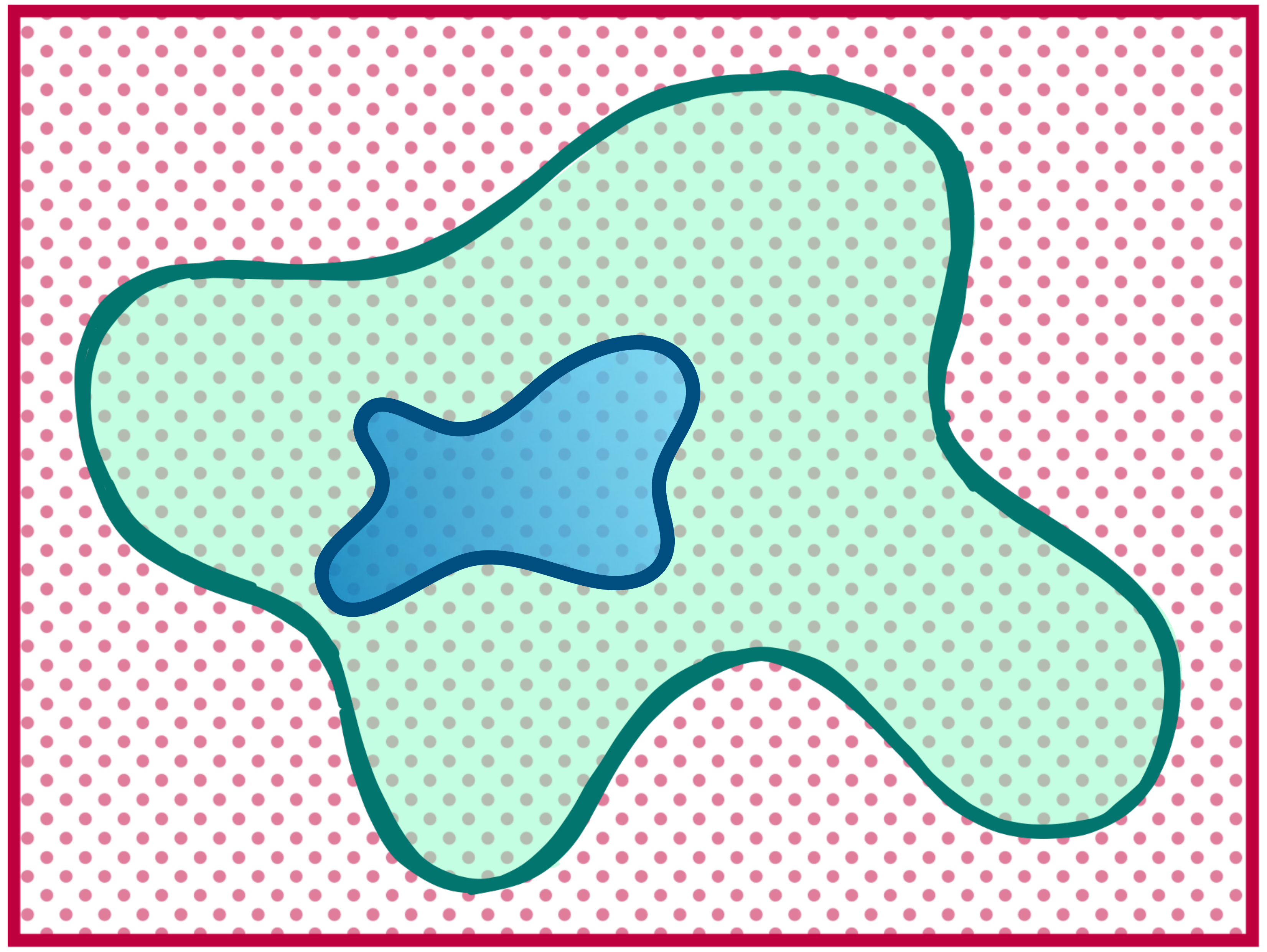
The class of “regular” spaghetti processes!

Reality

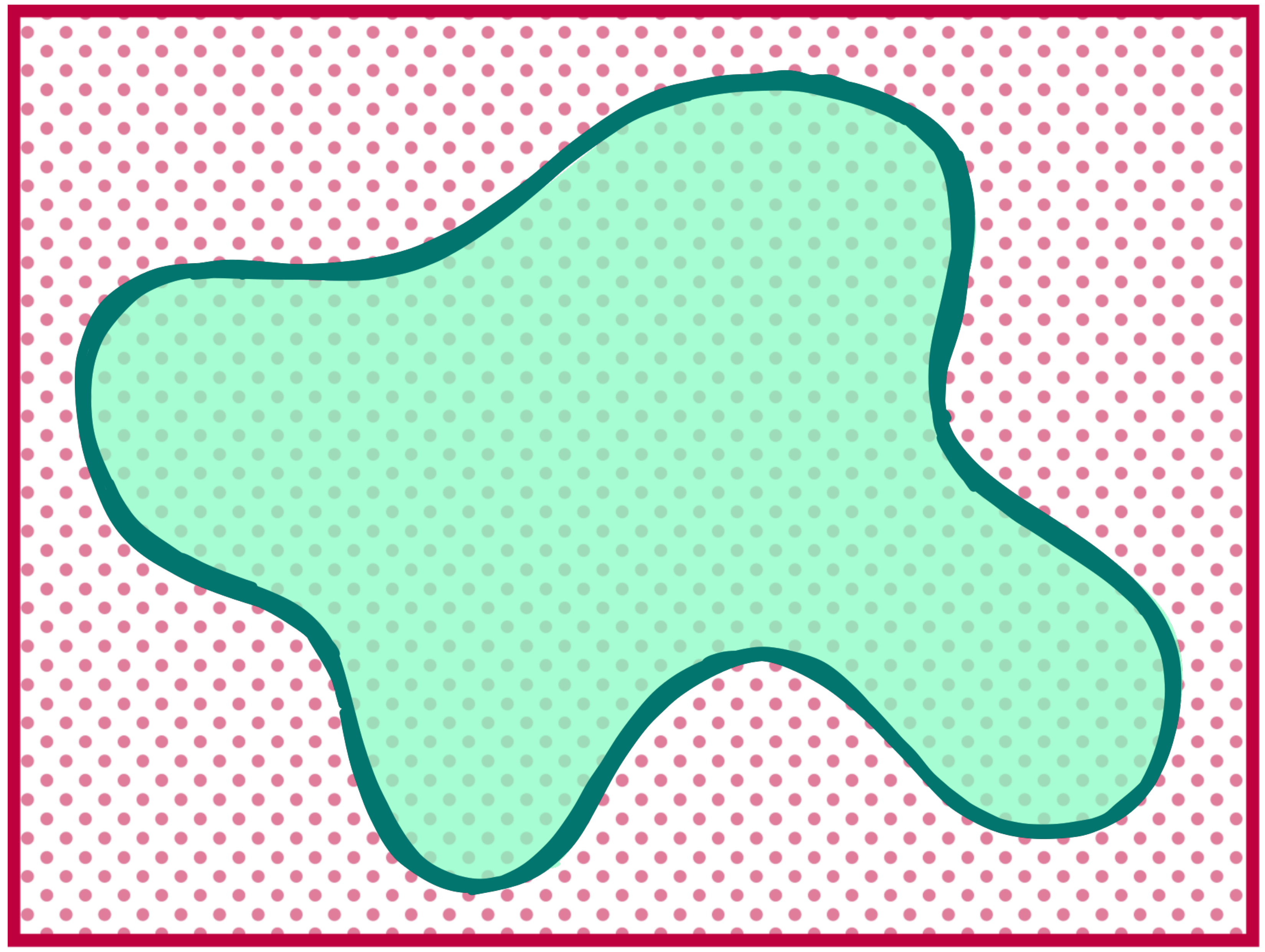
“Framing” via declarative specifications



Process



Imperative
model



Declarative
specification

“Framing” via declarative specifications

Process

Imperative model

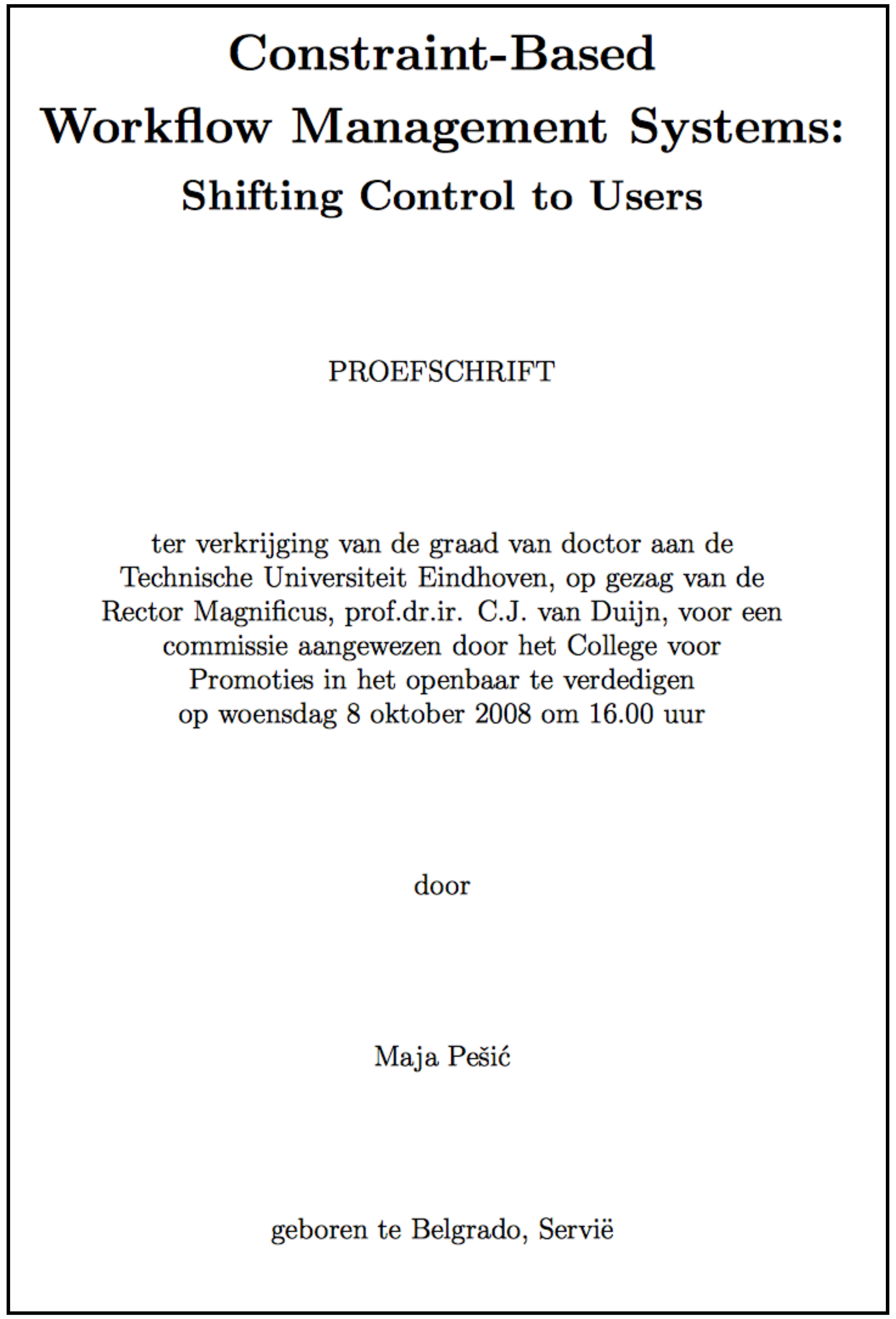
Declarative specification

Constraint-based specifications of behaviour

- **Multiagent systems**: declarative agent programs [Fisher,JSC1996] and interaction protocols [Singh,AAMAS2003]
- **Data management**: cascaded transactional updates [DavulcuEtAl,PODS1998]
- **BPM (1st wave)**: loosely-coupled subprocesses [SadiqEtAl,ER2001]
- **BPM (2nd wave)**: process constraints
  - **DECLARE** [PesicEtAl,EDOC2007]
  - Dynamic Condition-Response (DCR) Graphs [HildebrandtEtAl,PLACES2010]

Origin of Declare…

Language, formalisation, reasoning, enactment



Constraint-Based
Workflow Management Systems:
Shifting Control to Users

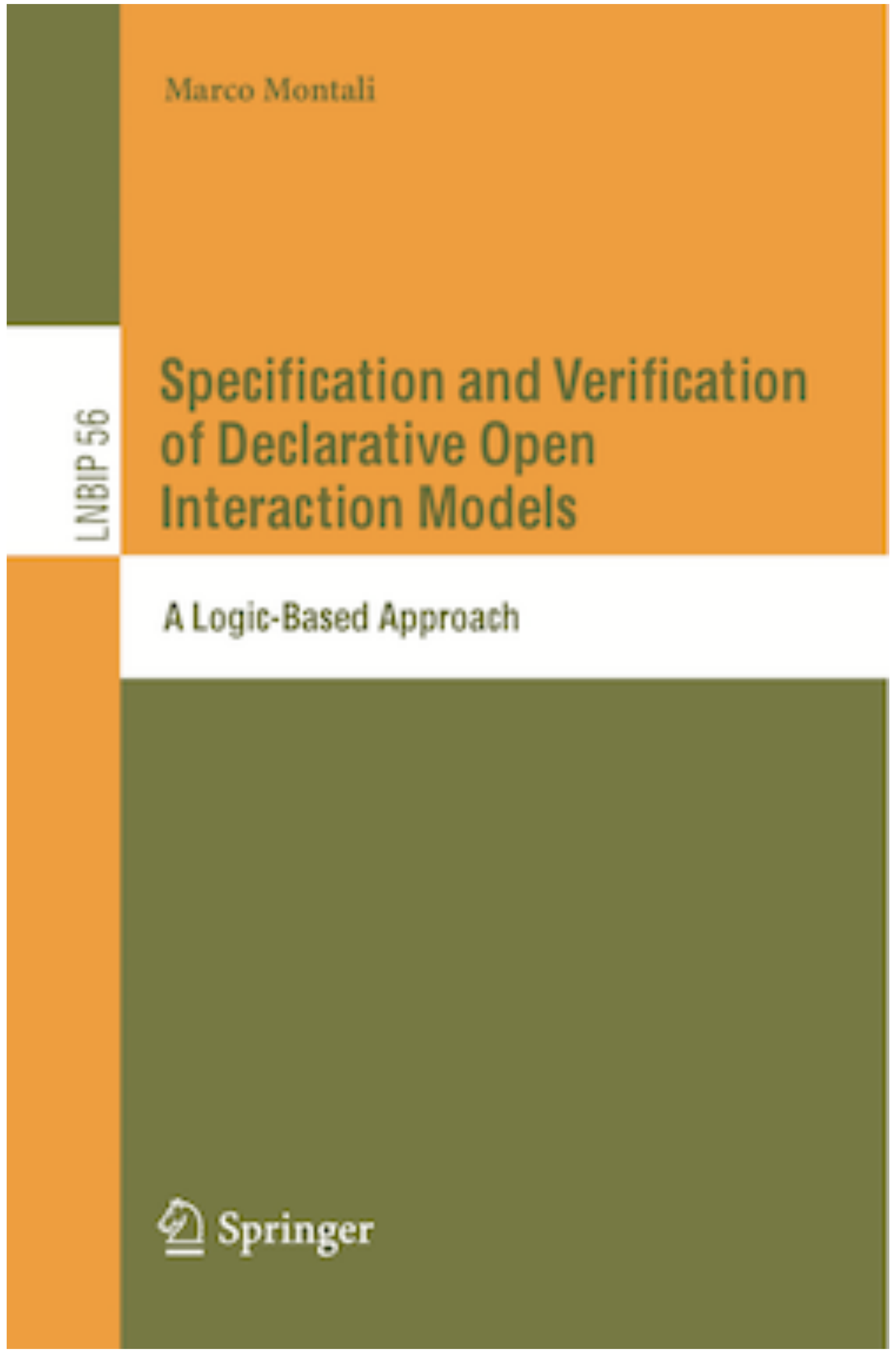
PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op woensdag 8 oktober 2008 om 16.00 uur

door

Maja Pešić

geboren te Belgrado, Servië



Marco Montali

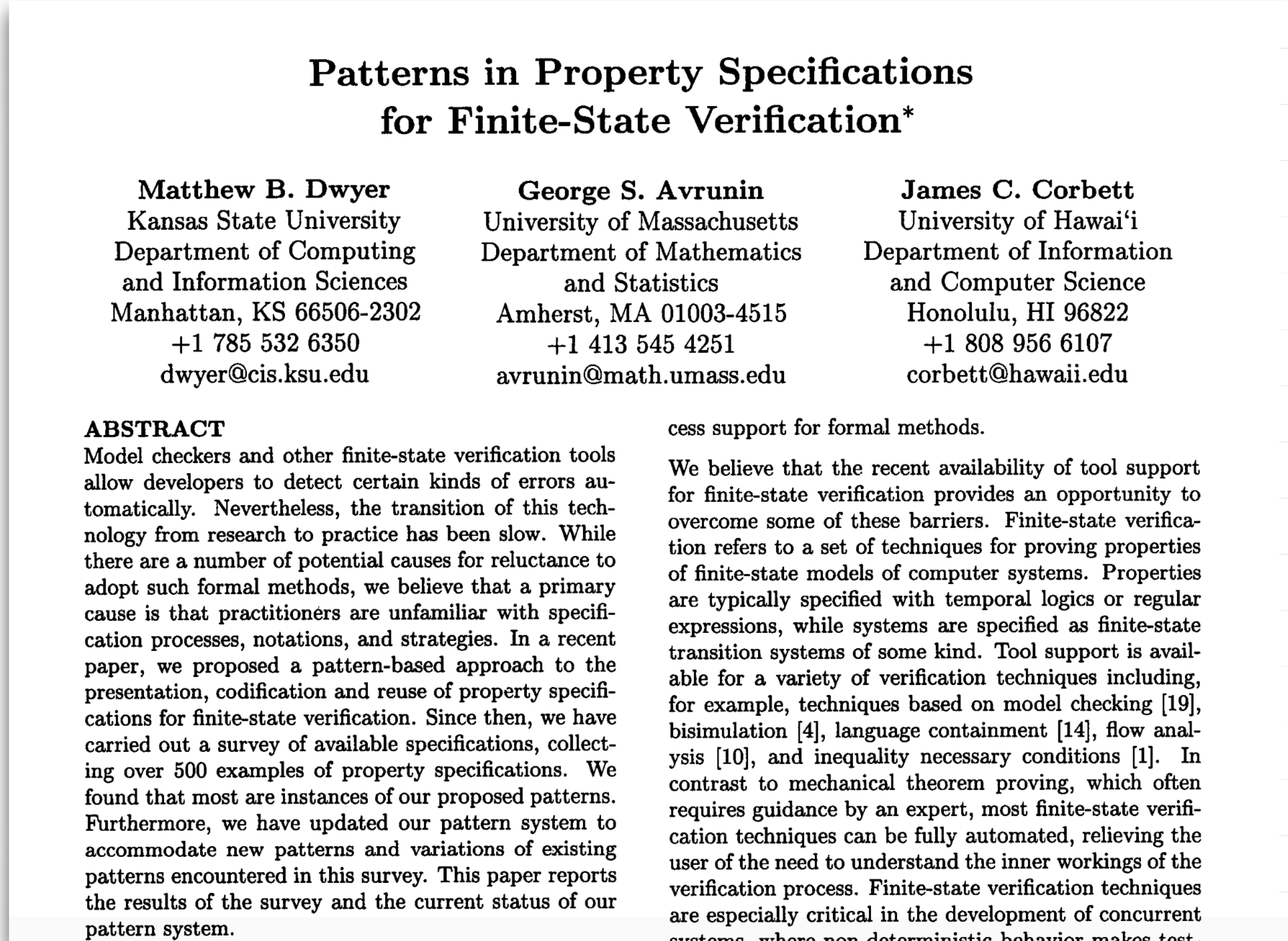
LNBIP 56

Specification and Verification of Declarative Open Interaction Models

A Logic-Based Approach

Springer

Which constraints are useful?



Patterns in Property Specifications for Finite-State Verification*

Matthew B. Dwyer
Kansas State University
Department of Computing
and Information Sciences
Manhattan, KS 66506-2302
+1 785 532 6350
dwyer@cis.ksu.edu

George S. Avrunin
University of Massachusetts
Department of Mathematics
and Statistics
Amherst, MA 01003-4515
+1 413 545 4251
avrunin@math.umass.edu

James C. Corbett
University of Hawai'i
Department of Information
and Computer Science
Honolulu, HI 96822
+1 808 956 6107
corbett@hawaii.edu

ABSTRACT

Model checkers and other finite-state verification tools allow developers to detect certain kinds of errors automatically. Nevertheless, the transition of this technology from research to practice has been slow. While there are a number of potential causes for reluctance to adopt such formal methods, we believe that a primary cause is that practitioners are unfamiliar with specification processes, notations, and strategies. In a recent paper, we proposed a pattern-based approach to the presentation, codification and reuse of property specifications for finite-state verification. Since then, we have carried out a survey of available specifications, collecting over 500 examples of property specifications. We found that most are instances of our proposed patterns. Furthermore, we have updated our pattern system to accommodate new patterns and variations of existing patterns encountered in this survey. This paper reports the results of the survey and the current status of our pattern system.

cess support for formal methods.

We believe that the recent availability of tool support for finite-state verification provides an opportunity to overcome some of these barriers. Finite-state verification refers to a set of techniques for proving properties of finite-state models of computer systems. Properties are typically specified with temporal logics or regular expressions, while systems are specified as finite-state transition systems of some kind. Tool support is available for a variety of verification techniques including, for example, techniques based on model checking [19], bisimulation [4], language containment [14], flow analysis [10], and inequality necessary conditions [1]. In contrast to mechanical theorem proving, which often requires guidance by an expert, most finite-state verification techniques can be fully automated, relieving the user of the need to understand the inner workings of the verification process. Finite-state verification techniques are especially critical in the development of concurrent systems, where non-deterministic behavior makes test-

Constraint templates

Constraint types defined on **activity placeholders**, each with a specific meaning

- … then **instantiated on actual activities** (by grounding)

Dimensions

- **Activities**: **how many** are involved
- **Time**: temporal **orientation** (past, future, either) and **strength** (how close)
- **Expectation**: **negative** vs **positive**

Much richer than the precedence flow relation of imperative languages

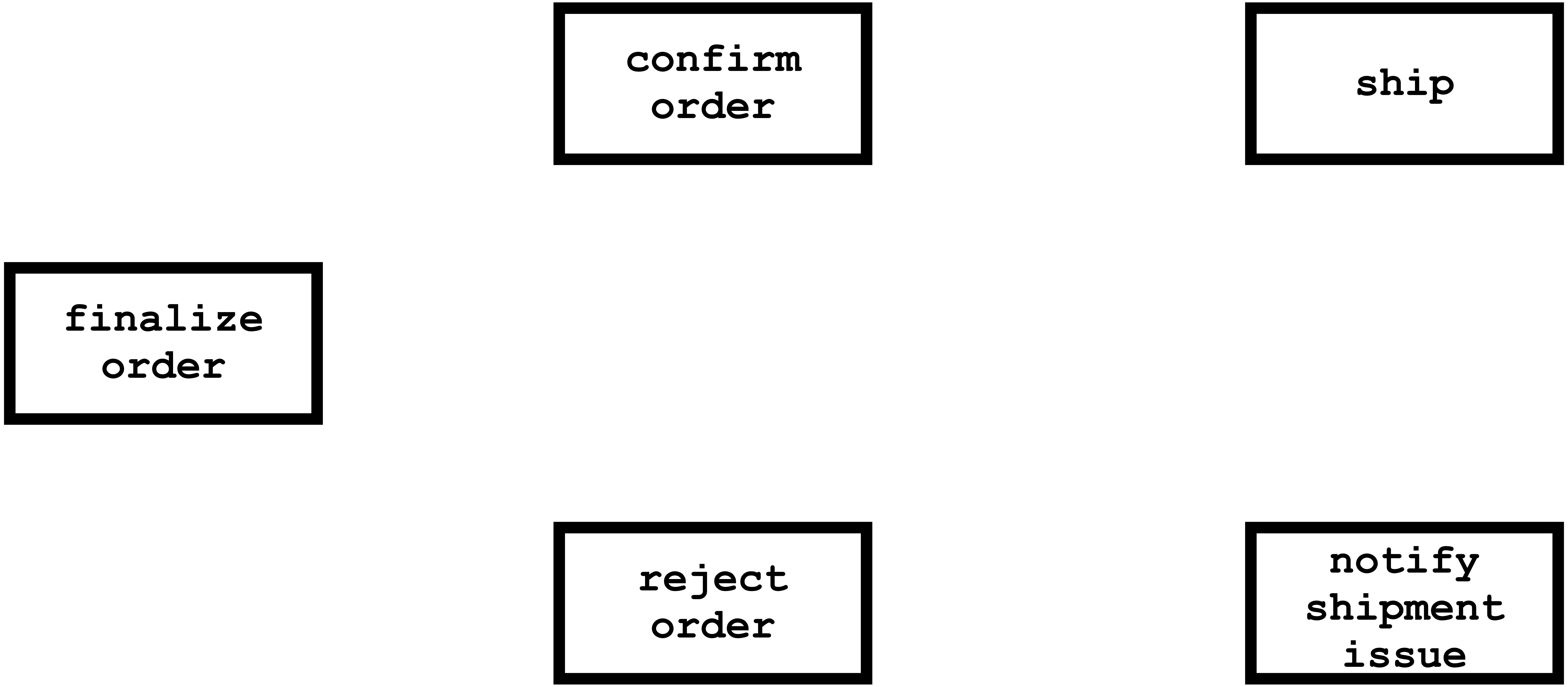
Declare specification

A set of constraints

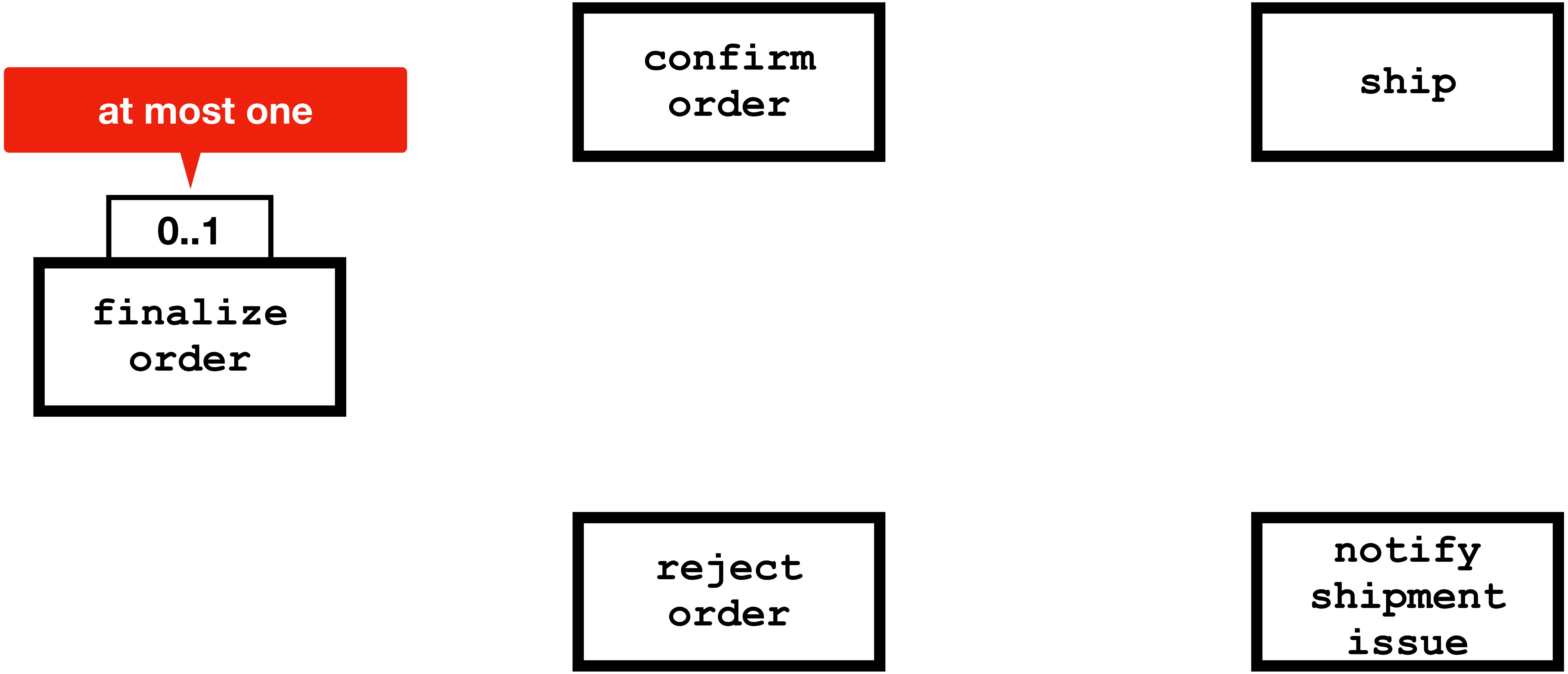
= templates grounded on the activities of interest

- Constraints have to be **all satisfied** over a **complete execution trace**
- Compositional approach by **conjunction**

Example


confirm order
ship
finalize order
reject order
notify shipment issue

Example



at most one

0..1

finalize order

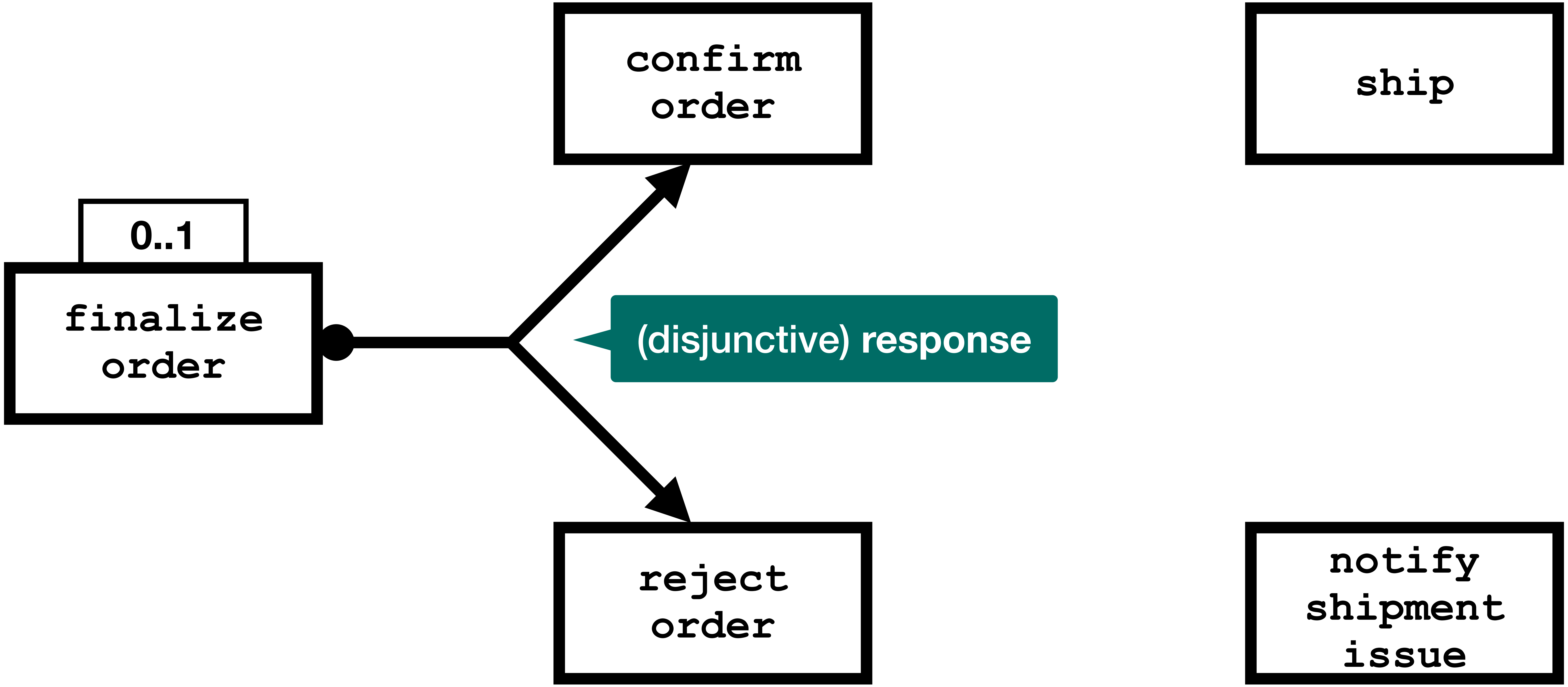
confirm order

ship

reject order

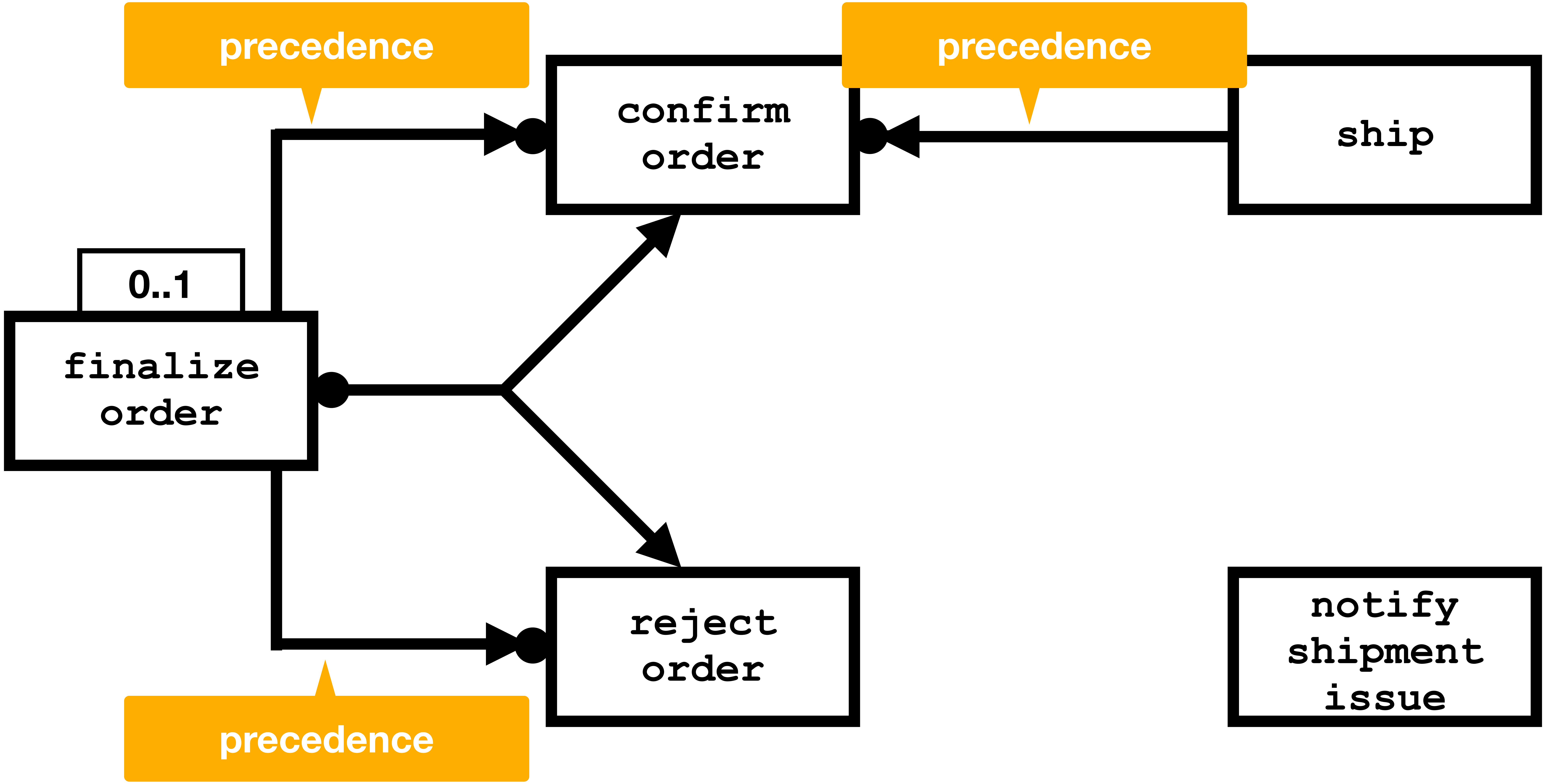
notify shipment issue

Example

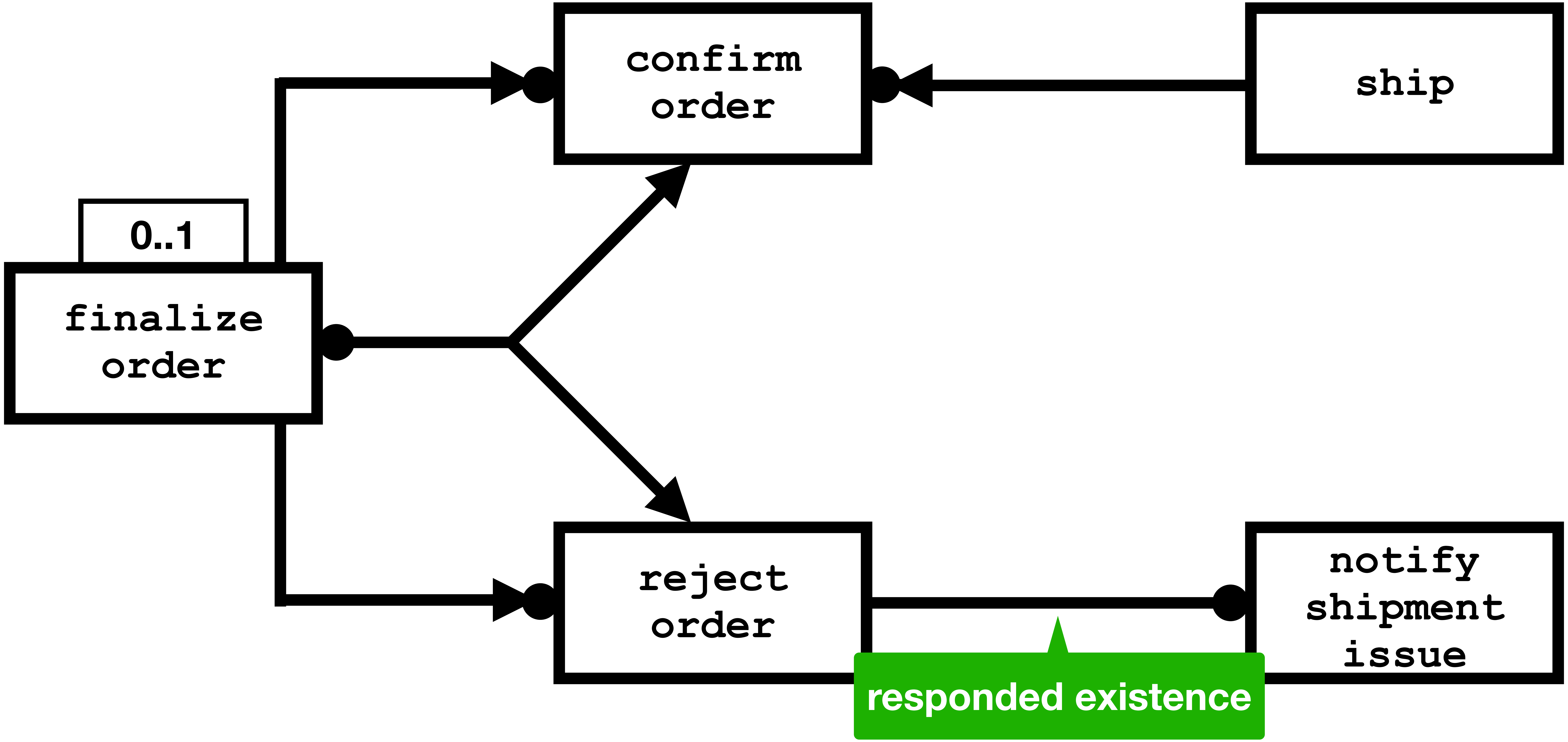


(disjunctive) response

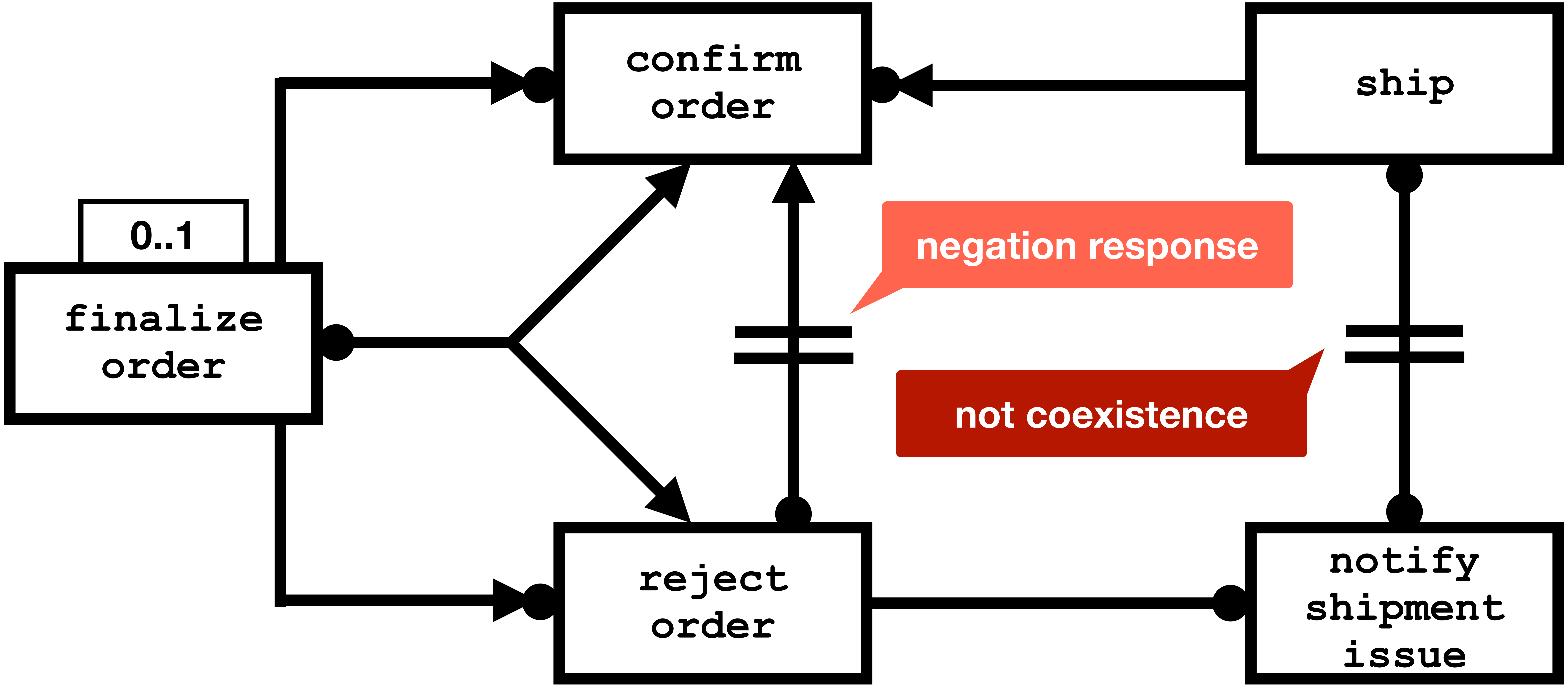
Example


precedence
precedence
confirm order
ship
0..1
finalize order
reject order
notify shipment issue
precedence

Example


confirm order
ship
0..1
finalize order
reject order
notify shipment issue
responded existence

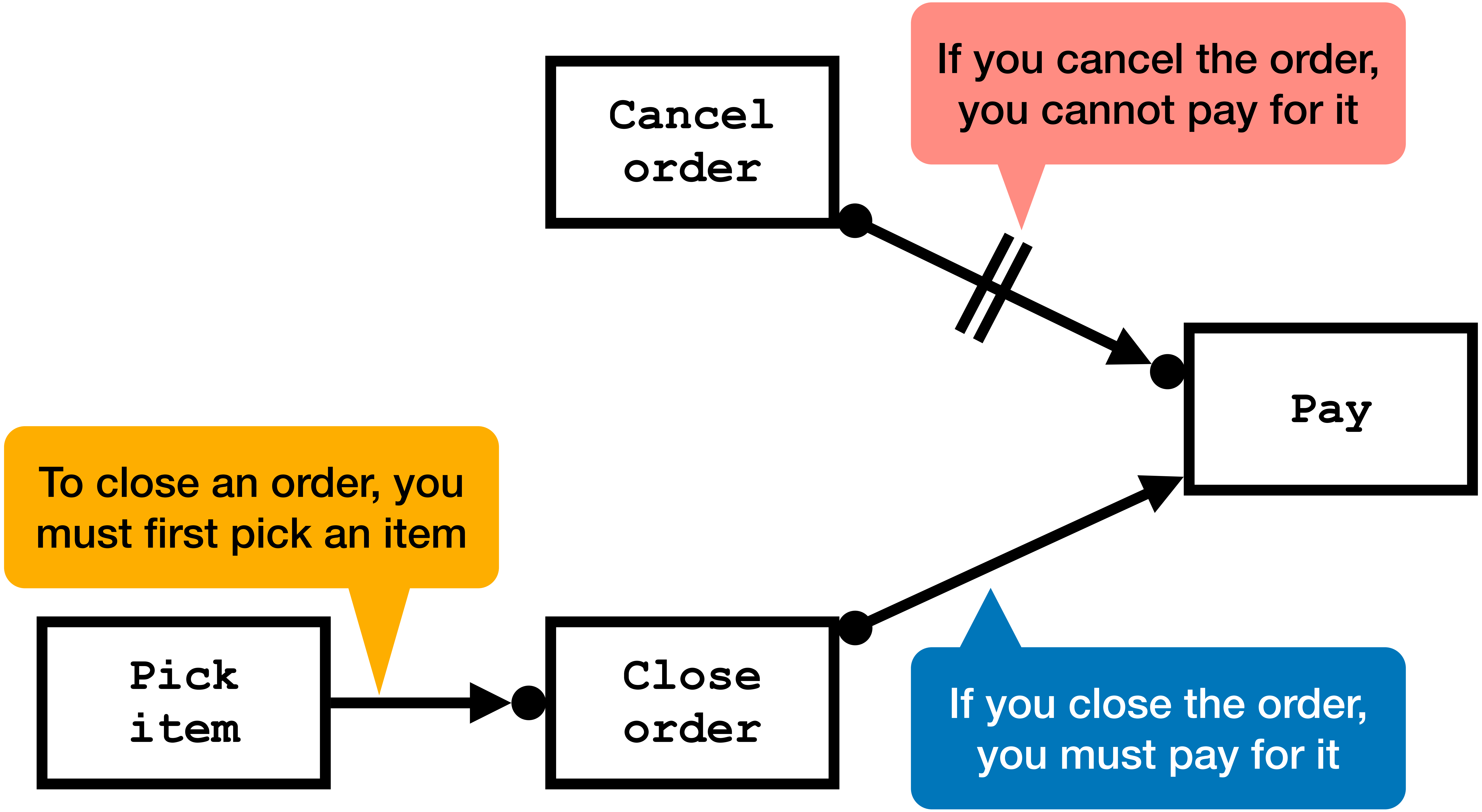
Example



0..1
finalize order
confirm order
ship
reject order
notify shipment issue
negation response
not coexistence

Interaction among constraints

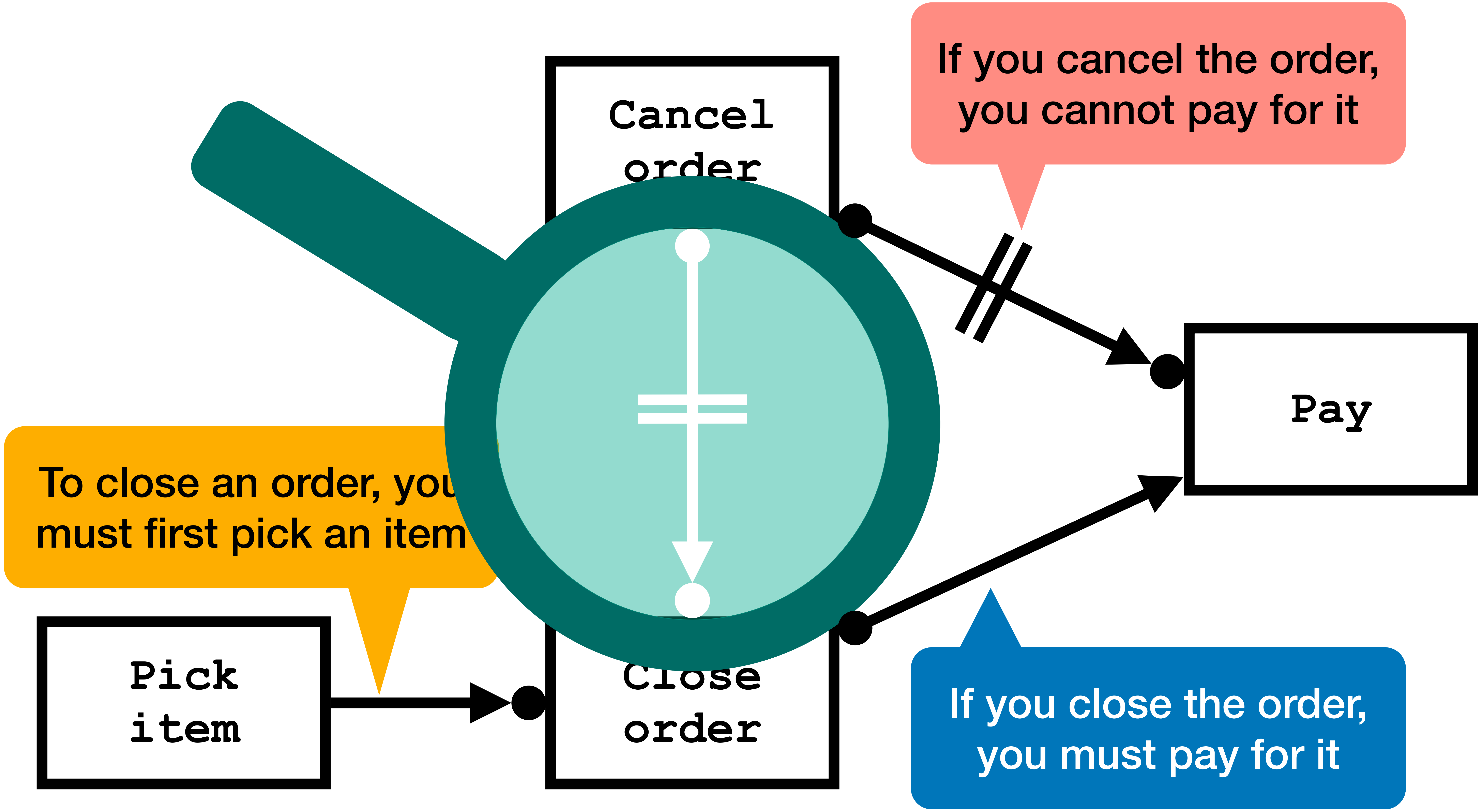
Aka hidden dependencies [____,TWEB2010]



Cancel order
If you cancel the order, you cannot pay for it
Pay
To close an order, you must first pick an item
Pick item
Close order
If you close the order, you must pay for it

Interaction among constraints

Aka hidden dependencies [____,TWEB2010]

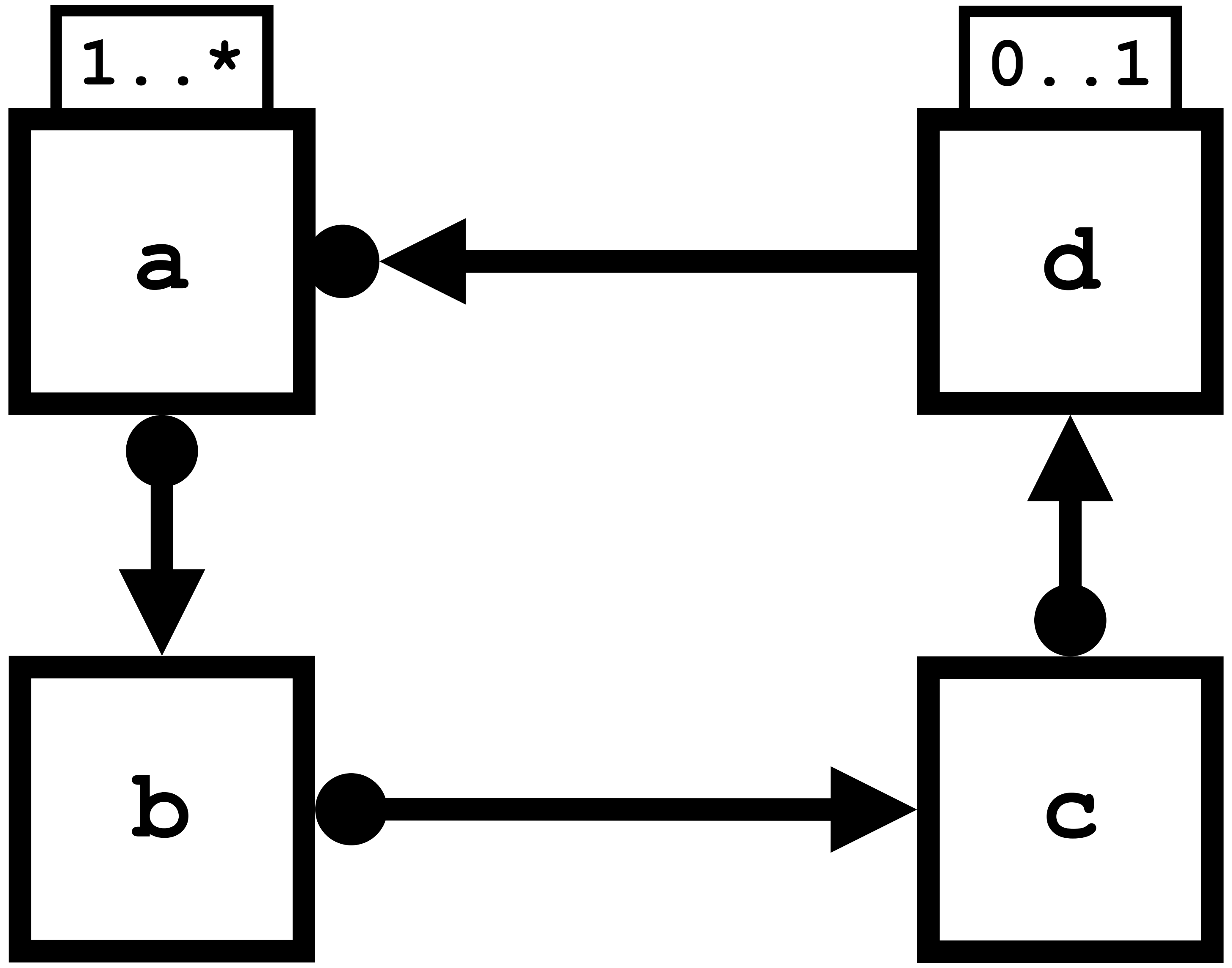


If you cancel the order, you cannot pay for it

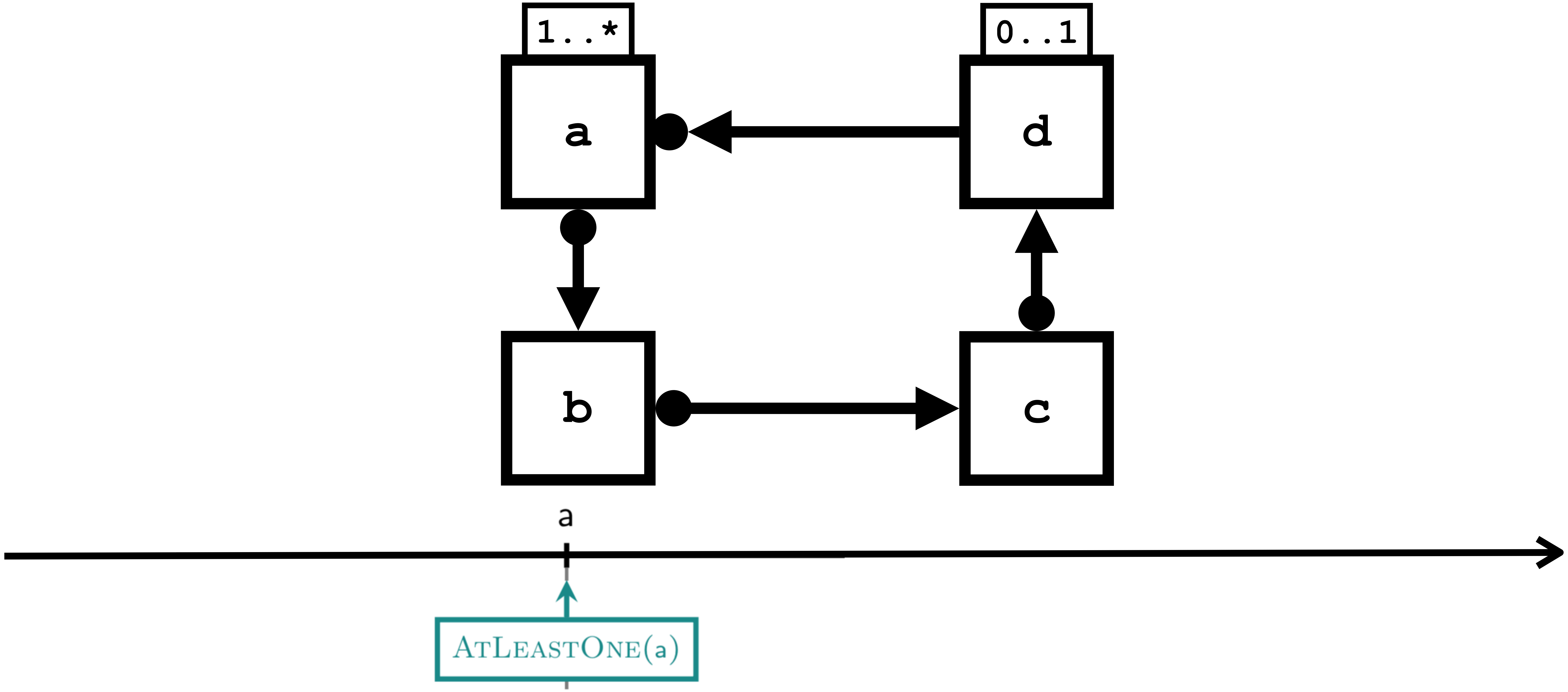
To close an order, you must first pick an item

If you close the order, you must pay for it

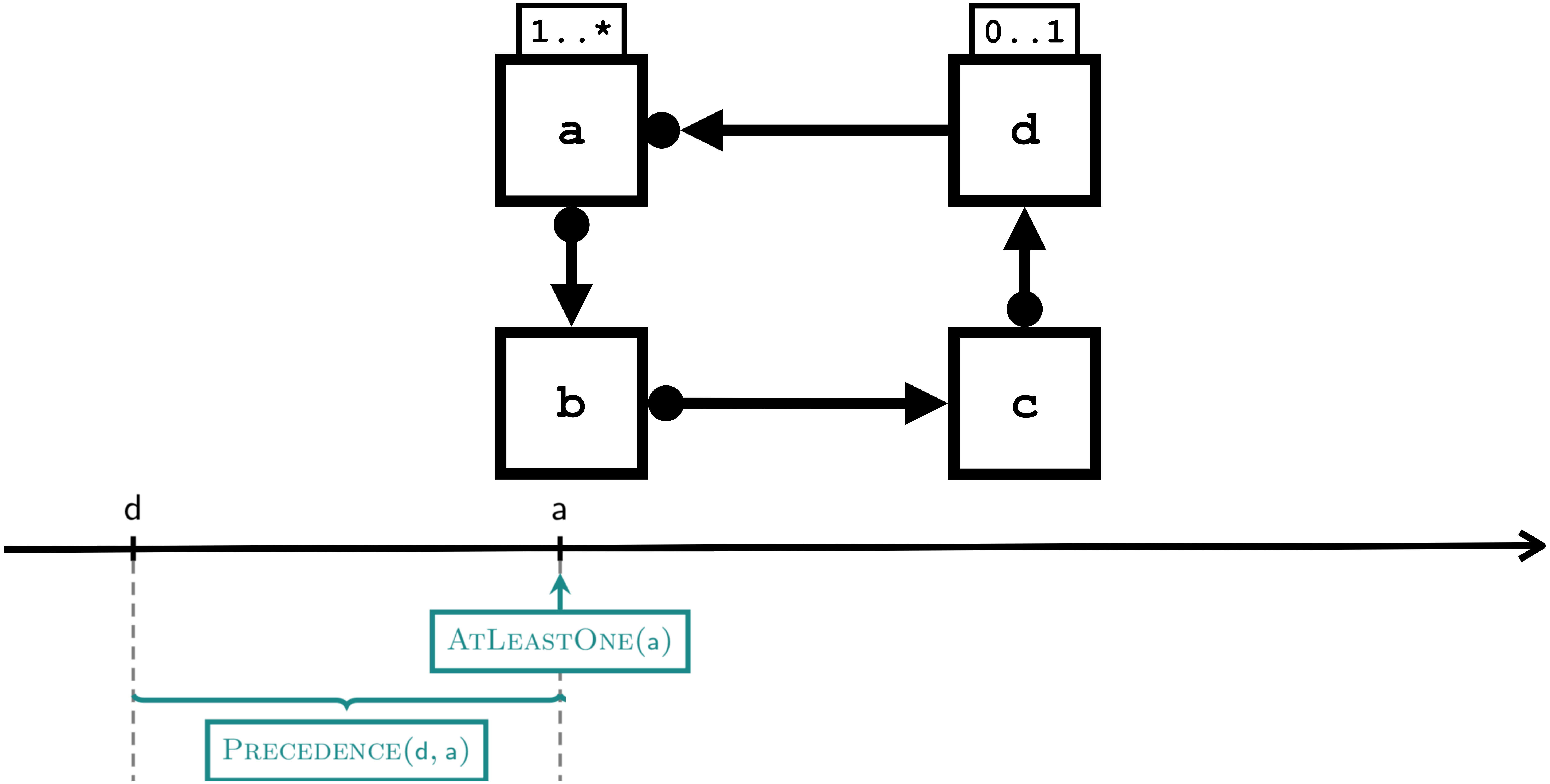
Quiz: does this specification accept traces?



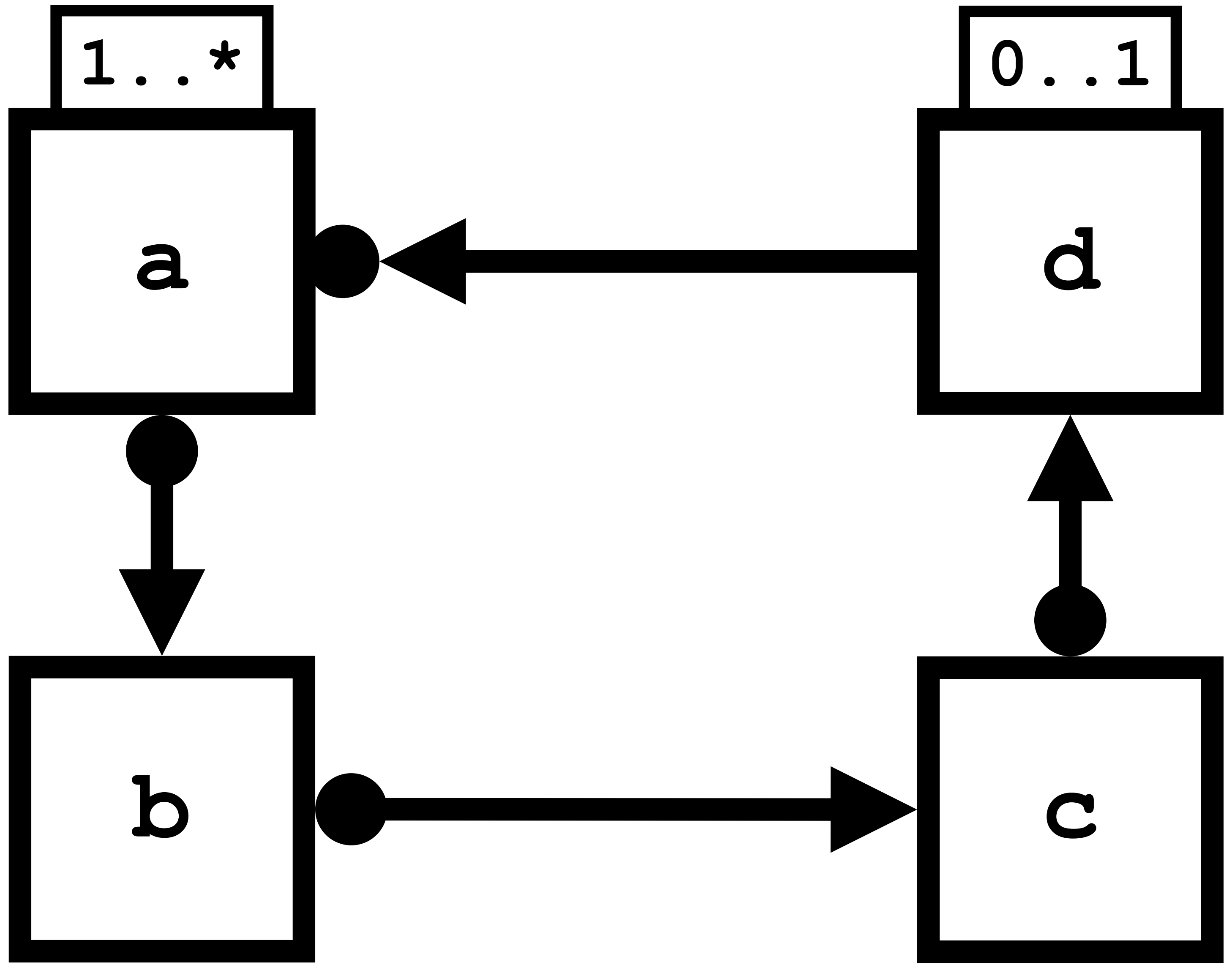
Quiz: does this specification accept traces?

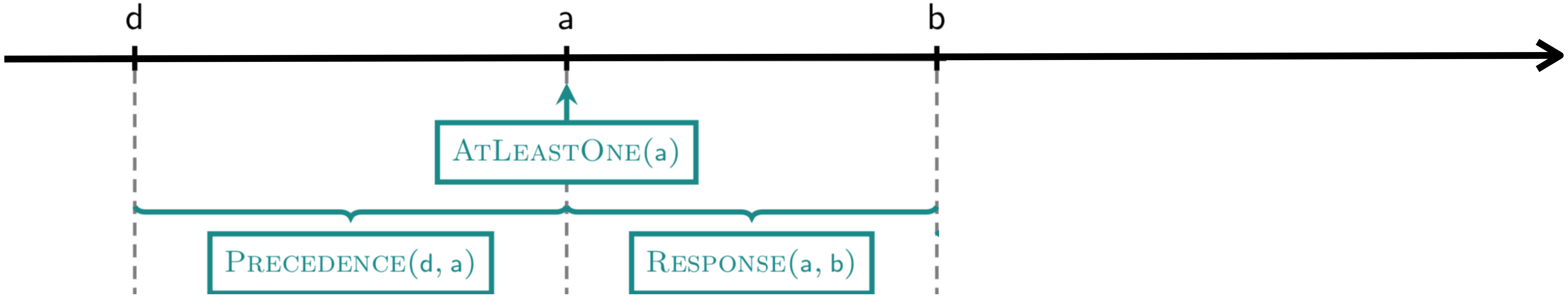


Quiz: does this specification accept traces?

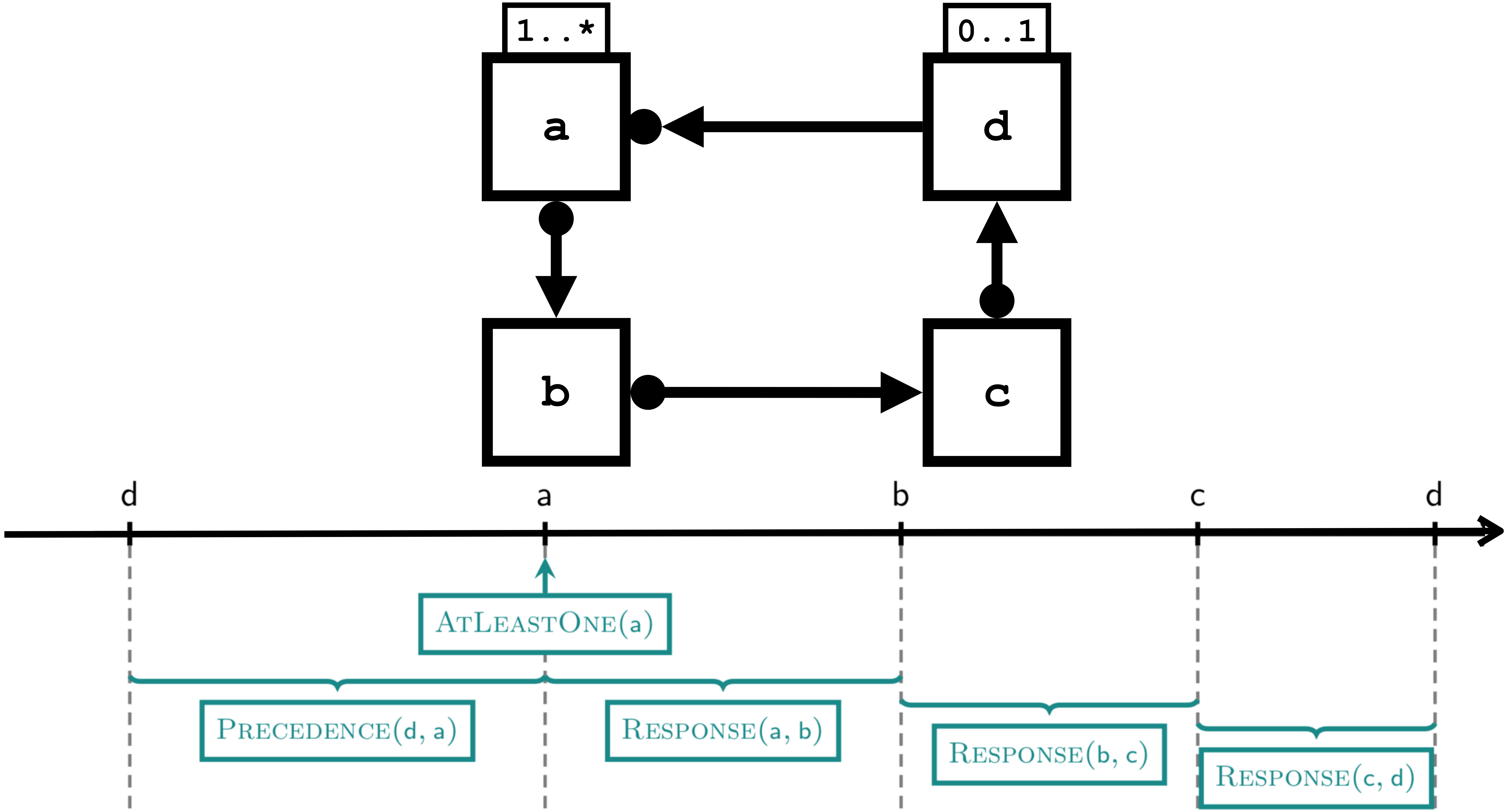


Quiz: does this specification accept traces?

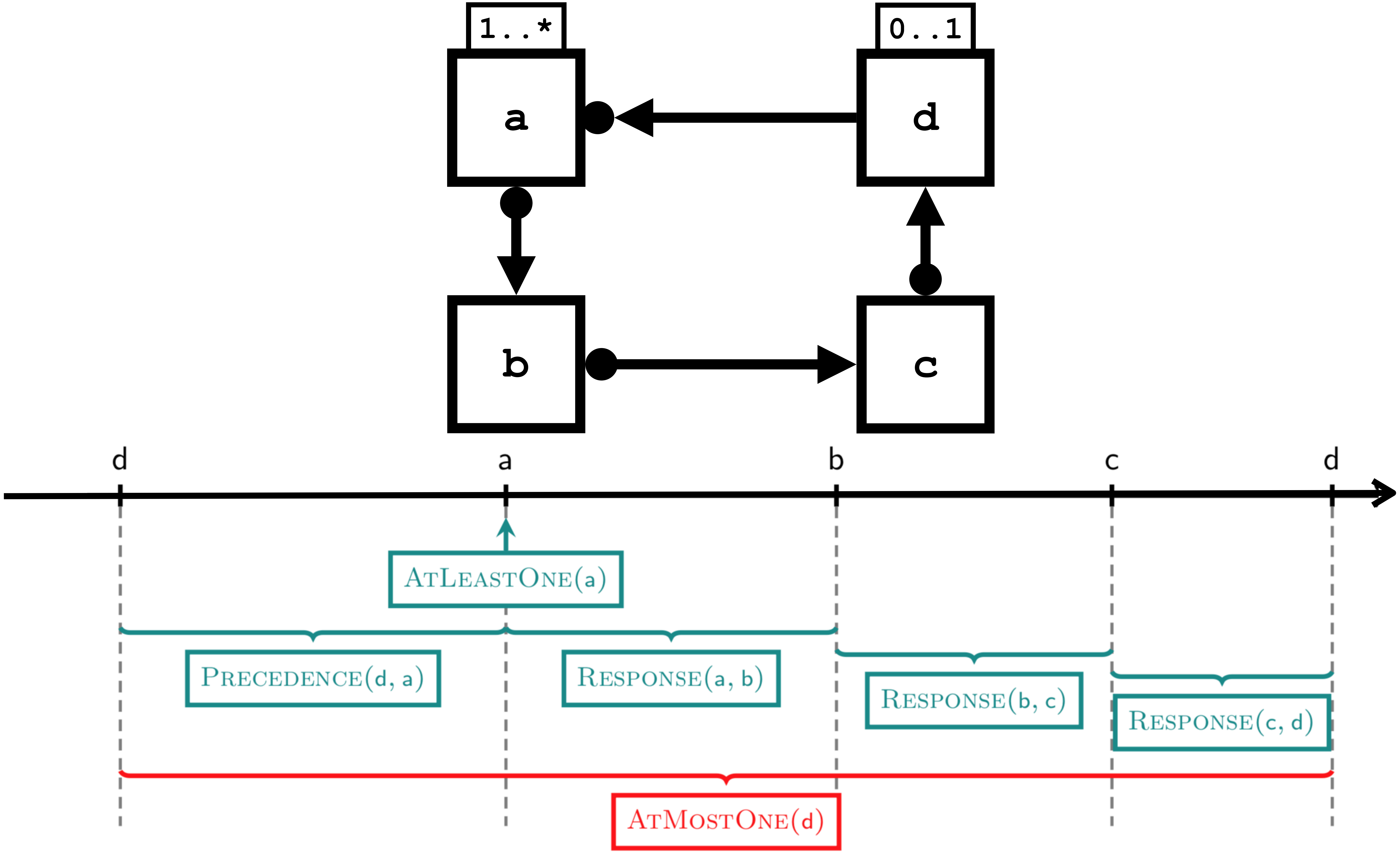




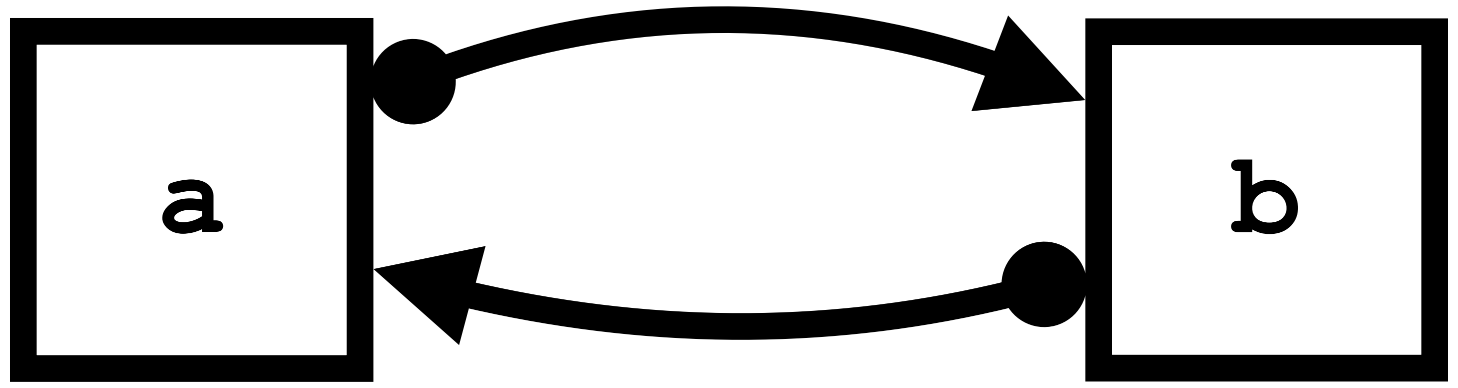
Quiz: does this specification accept traces?



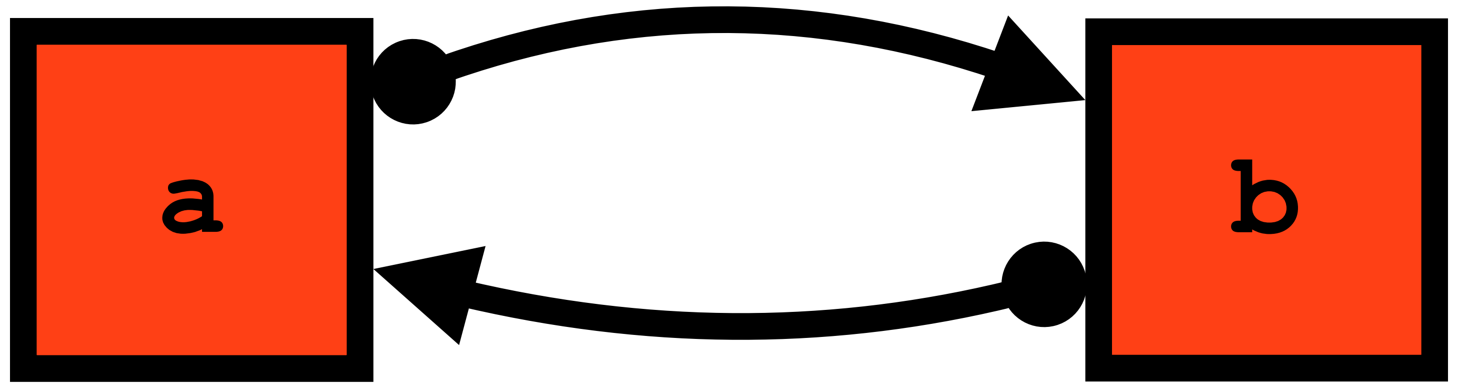
Quiz: does this specification accept traces?



Quiz: does this specification accept traces?



Quiz: does this specification accept traces?



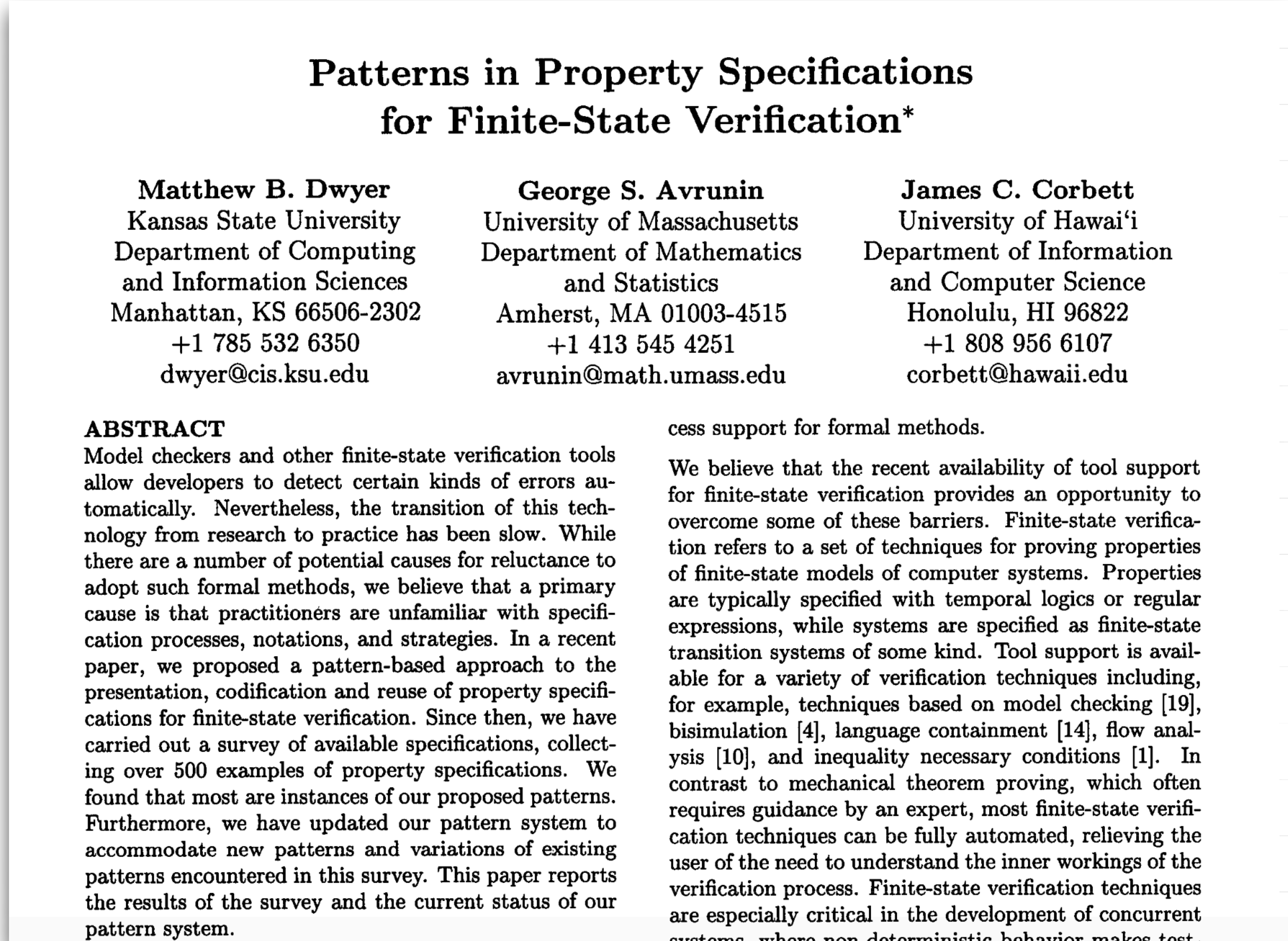
Only the empty trace <>, due to finite-trace semantics

How to understand if a Declare specification is correct?

How to characterise the traces of a Declare specification?

LTLf and automata to the rescue

Back to the roots



Patterns in Property Specifications for Finite-State Verification*

Matthew B. Dwyer
Kansas State University
Department of Computing
and Information Sciences
Manhattan, KS 66506-2302
+1 785 532 6350
dwyer@cis.ksu.edu

George S. Avrunin
University of Massachusetts
Department of Mathematics
and Statistics
Amherst, MA 01003-4515
+1 413 545 4251
avrunin@math.umass.edu

James C. Corbett
University of Hawai'i
Department of Information
and Computer Science
Honolulu, HI 96822
+1 808 956 6107
corbett@hawaii.edu

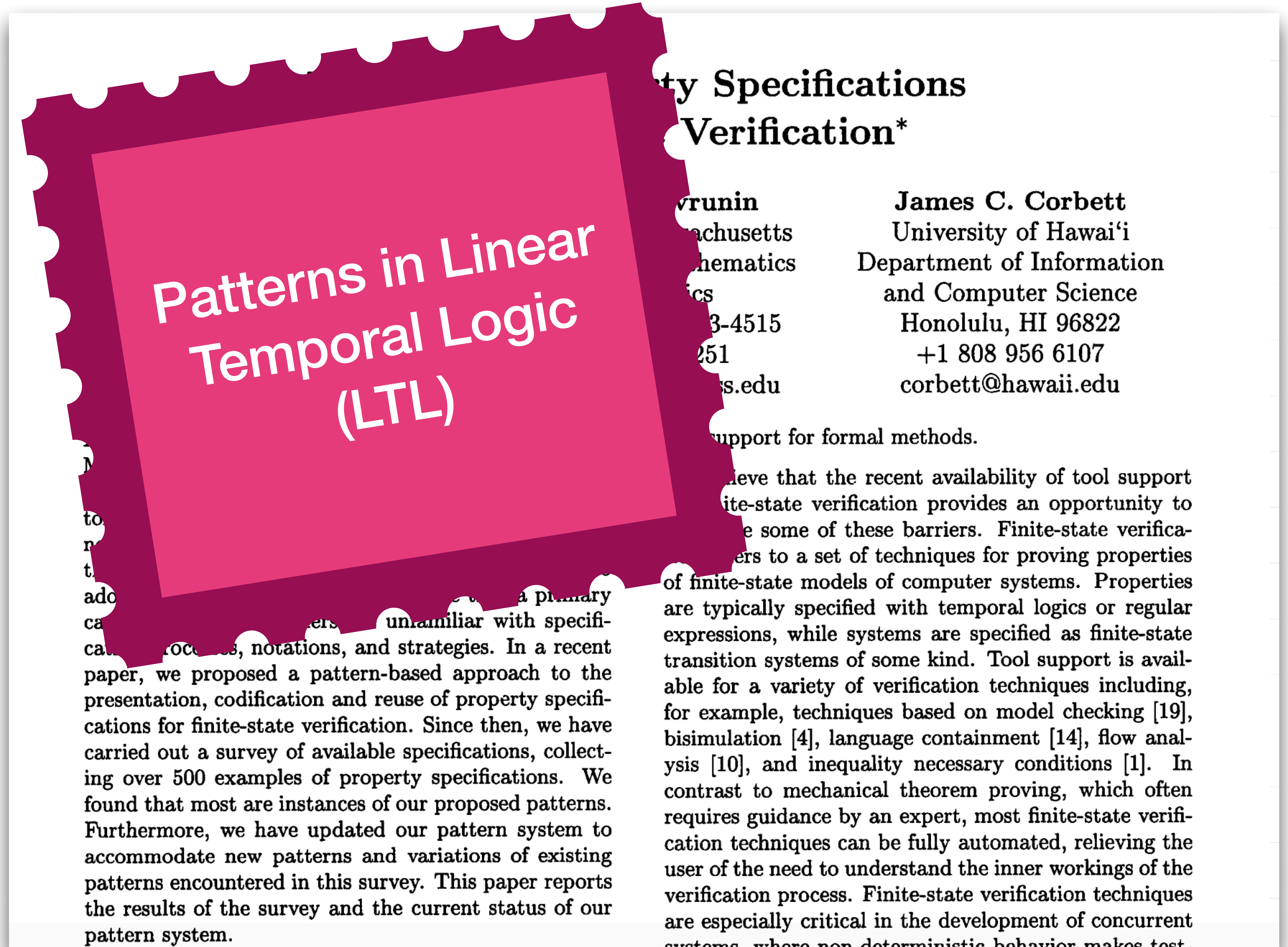
ABSTRACT

Model checkers and other finite-state verification tools allow developers to detect certain kinds of errors automatically. Nevertheless, the transition of this technology from research to practice has been slow. While there are a number of potential causes for reluctance to adopt such formal methods, we believe that a primary cause is that practitioners are unfamiliar with specification processes, notations, and strategies. In a recent paper, we proposed a pattern-based approach to the presentation, codification and reuse of property specifications for finite-state verification. Since then, we have carried out a survey of available specifications, collecting over 500 examples of property specifications. We found that most are instances of our proposed patterns. Furthermore, we have updated our pattern system to accommodate new patterns and variations of existing patterns encountered in this survey. This paper reports the results of the survey and the current status of our pattern system.

cess support for formal methods.

We believe that the recent availability of tool support for finite-state verification provides an opportunity to overcome some of these barriers. Finite-state verification refers to a set of techniques for proving properties of finite-state models of computer systems. Properties are typically specified with temporal logics or regular expressions, while systems are specified as finite-state transition systems of some kind. Tool support is available for a variety of verification techniques including, for example, techniques based on model checking [19], bisimulation [4], language containment [14], flow analysis [10], and inequality necessary conditions [1]. In contrast to mechanical theorem proving, which often requires guidance by an expert, most finite-state verification techniques can be fully automated, relieving the user of the need to understand the inner workings of the verification process. Finite-state verification techniques are especially critical in the development of concurrent systems, where non-deterministic behavior makes test-

Back to the roots



Patterns in Linear Temporal Logic (LTL)

ty Specifications
Verification*

vrunin
achusetts
hematics
ics
3-4515
251
s.edu

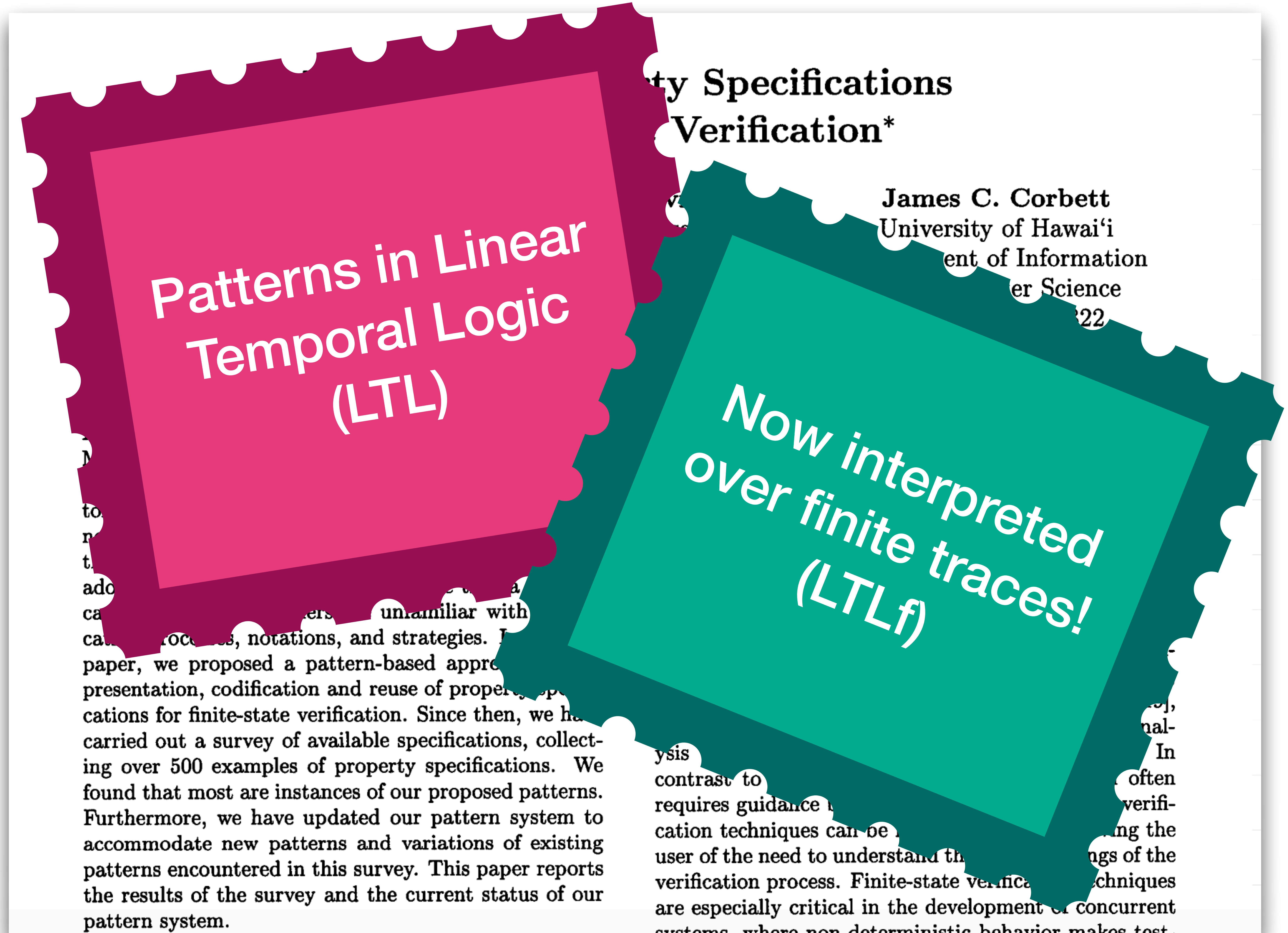
James C. Corbett
University of Hawai'i
Department of Information
and Computer Science
Honolulu, HI 96822
+1 808 956 6107
corbett@hawaii.edu

upport for formal methods.

to
n
t
ado a primary
ca ers unfamiliar with specifi-
ca roc s, notations, and strategies. In a recent paper, we proposed a pattern-based approach to the presentation, codification and reuse of property specifications for finite-state verification. Since then, we have carried out a survey of available specifications, collecting over 500 examples of property specifications. We found that most are instances of our proposed patterns. Furthermore, we have updated our pattern system to accommodate new patterns and variations of existing patterns encountered in this survey. This paper reports the results of the survey and the current status of our pattern system.

ieve that the recent availability of tool support ite-state verification provides an opportunity to e some of these barriers. Finite-state verifica- ers to a set of techniques for proving properties of finite-state models of computer systems. Properties are typically specified with temporal logics or regular expressions, while systems are specified as finite-state transition systems of some kind. Tool support is available for a variety of verification techniques including, for example, techniques based on model checking [19], bisimulation [4], language containment [14], flow analysis [10], and inequality necessary conditions [1]. In contrast to mechanical theorem proving, which often requires guidance by an expert, most finite-state verification techniques can be fully automated, relieving the user of the need to understand the inner workings of the verification process. Finite-state verification techniques are especially critical in the development of concurrent systems, where non-deterministic behavior makes test-

Back to the roots



ty Specifications

Verification*

James C. Corbett
University of Hawai'i
ent of Information
er Science
22

unfamiliar with ... notations, and strategies. In ... paper, we proposed a pattern-based appro... presentation, codification and reuse of property spe... cations for finite-state verification. Since then, we ha... carried out a survey of available specifications, collecting over 500 examples of property specifications. We found that most are instances of our proposed patterns. Furthermore, we have updated our pattern system to accommodate new patterns and variations of existing patterns encountered in this survey. This paper reports the results of the survey and the current status of our pattern system.

nal- ysis ... In contrast to ... often requires guidance ... verification techniques can be ...ng the user of the need to understand th...ngs of the verification process. Finite-state veri...chniques are especially critical in the development of concurrent systems, where non-deterministic behavior makes test

Patterns in Linear Temporal Logic (LTL)

Now interpreted over finite traces! (LTLf)

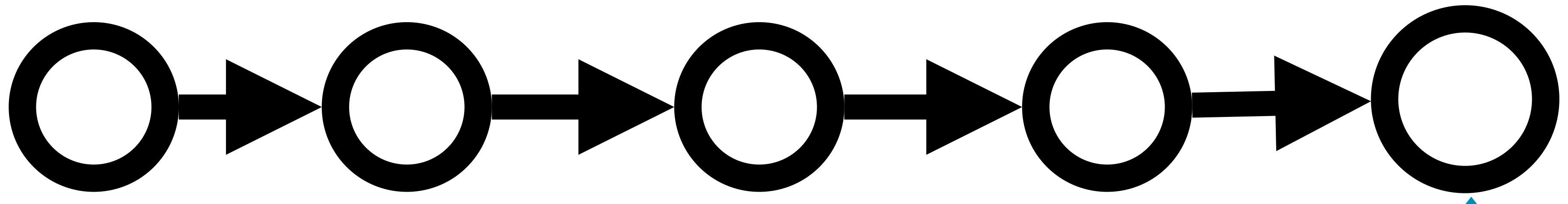
LTLf: LTL over finite traces

[DeGiacomoVardi,IJCAI2013]

$$\varphi ::= A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

Same syntax of LTL

LTL interpreted over **finite** traces



No successor!

In LTL, there is always a next moment… in LTLf, the contrary!

$\Box\varphi$	ϕ **always** holds from current **to the last** instant
$\bigcirc\varphi$	The **next step exists** and at **next** step ϕ holds
$\bullet\varphi \equiv \neg \bigcirc \neg\varphi$	(*weak next)* If the **next step exists**, then at **next** step ϕ holds
$Last \equiv \neg \bigcirc true$	**last** instant in the trace

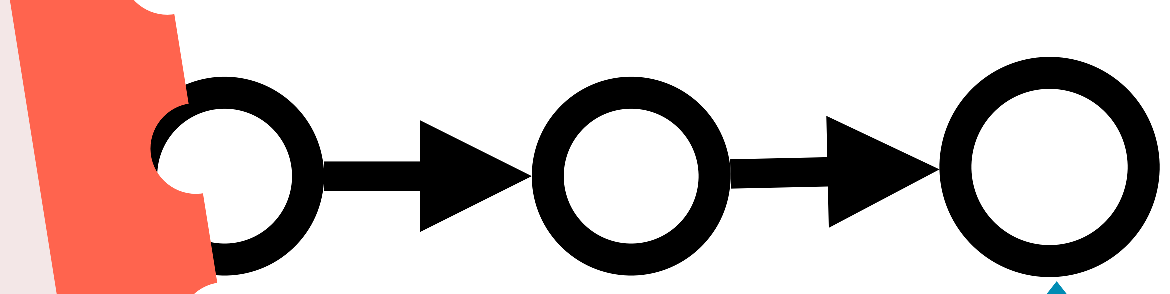
LTLf: LTL over finite traces

[DeGiacomoVardi,IJCAI2013]

$$\varphi ::= A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

Same syntax of LTL

LTL interpreted over **finit**



No successor!

In LTL, there is always a nex y!

Trace *t* satisfies a formula φ now formally defined:

$$t \models \varphi$$

$\Box\varphi$	ϕ a e **last** instant
$\bigcirc\varphi$	The step ϕ holds
$\bullet\varphi \equiv \neg\bigcirc\neg\varphi$	(*wea*) **next step exists**, then at **next** step ϕ holds
$Last \equiv \neg\bigcirc true$	**last** instant in the trace

Template formulae

LTLf

at-least-one(a)

at-most-one(a)

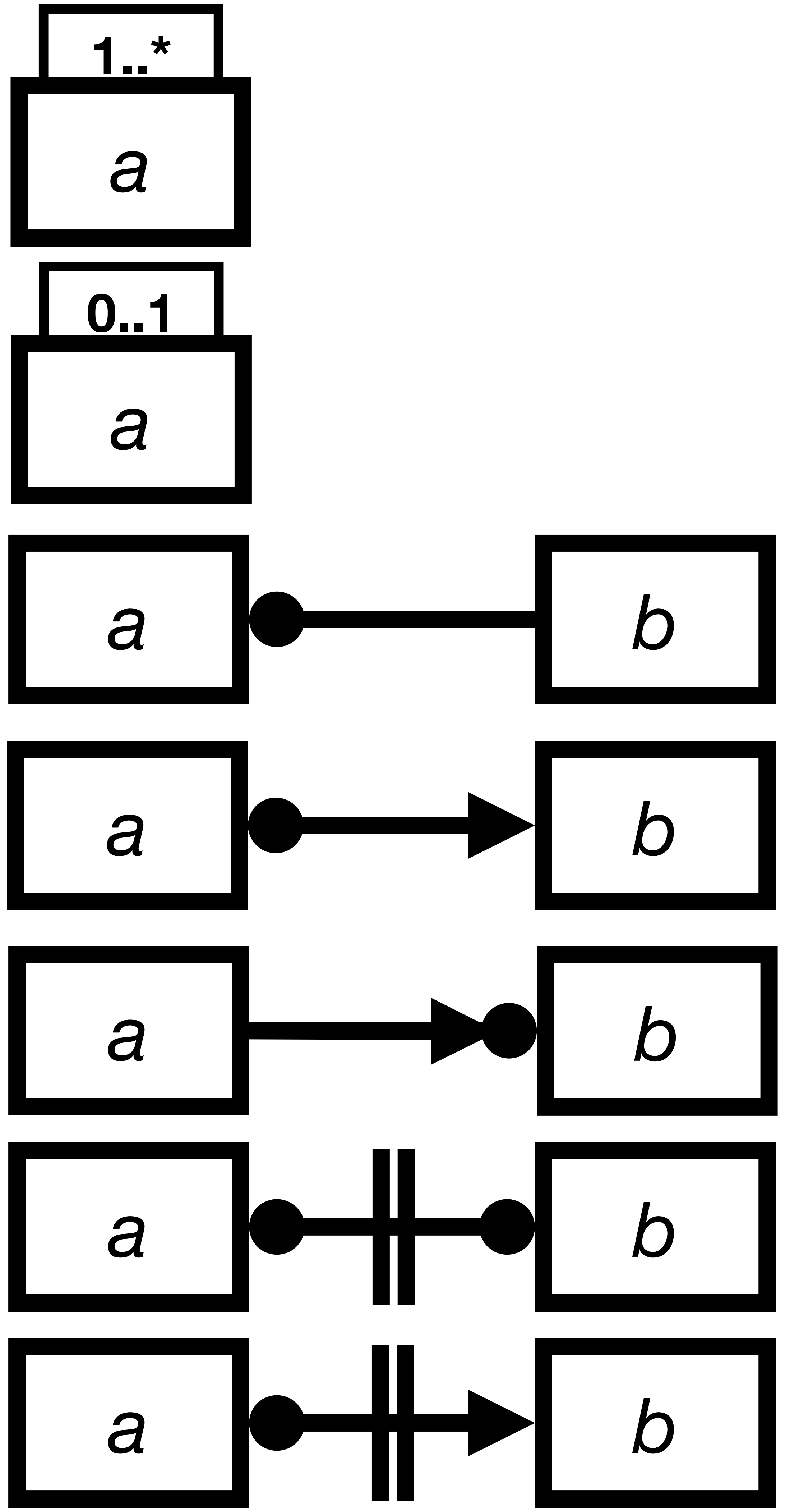
responded-existence(a,b)

response(a,b)

precedence(a,b)

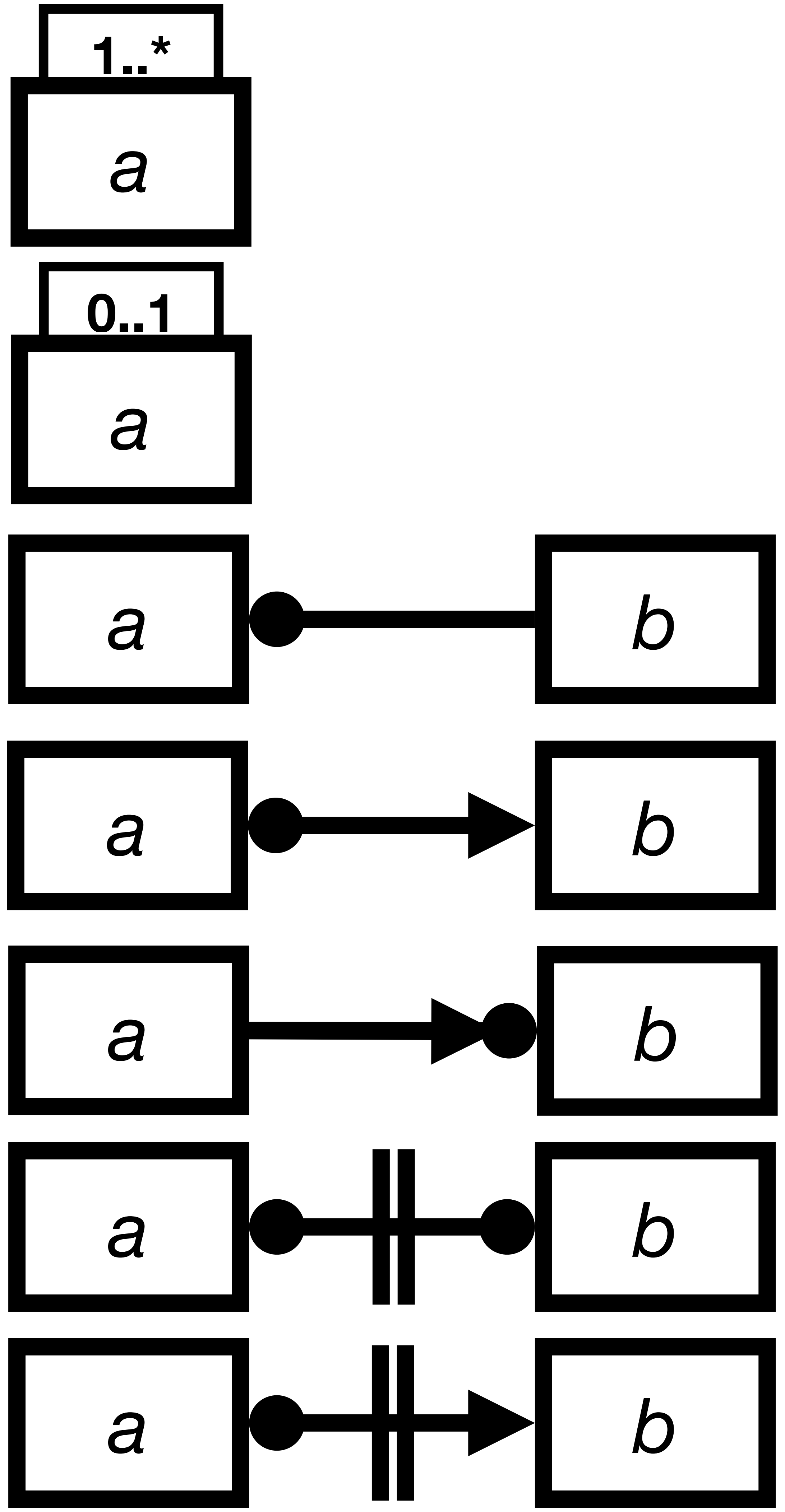
not-coexistence(a,b)

negation-response(a,b)



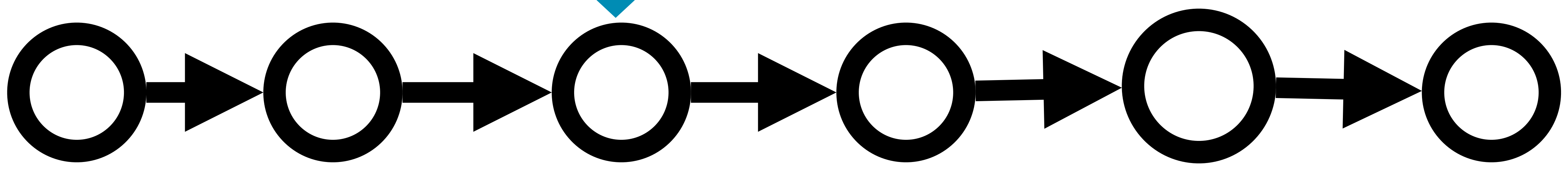
Template formulae

LTLf



at-least-one(a)	$\Diamond a$
at-most-one(a)	$\neg\Diamond(a \land \bigcirc\Diamond a)$
responded-existence(a,b)	$\Diamond a \rightarrow \Diamond b$
response(a,b)	$\Box(a \rightarrow \bigcirc\Diamond b)$
precedence(a,b)	$\neg(b \,\mathsf{W}\, a)$
not-coexistence(a,b)	$\neg(\Diamond a \land \Diamond b)$
negation-response(a,b)	$\Box(a \rightarrow \neg\bigcirc\Diamond b)$

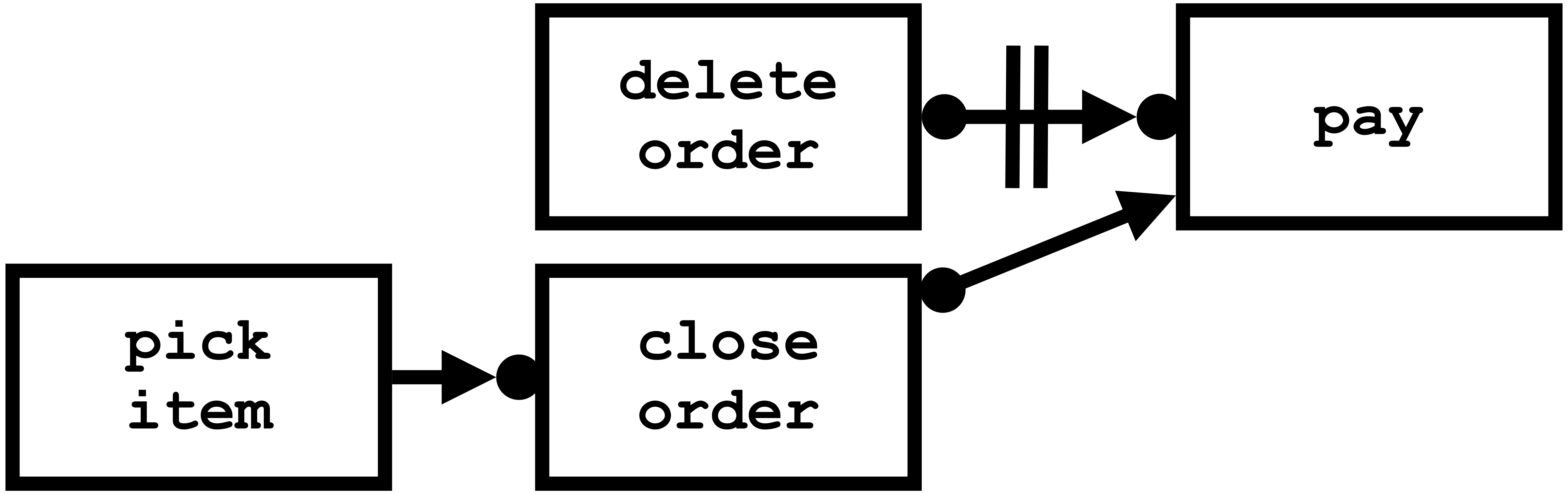
Semantics of Declare via LTLf



Each state contains a single activity

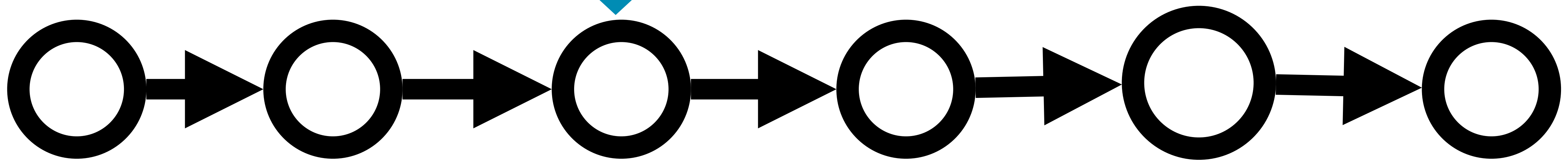
Atomic **propositions** are **activities**

Each **constraint** is an **LTL formula** (built from template formulae)



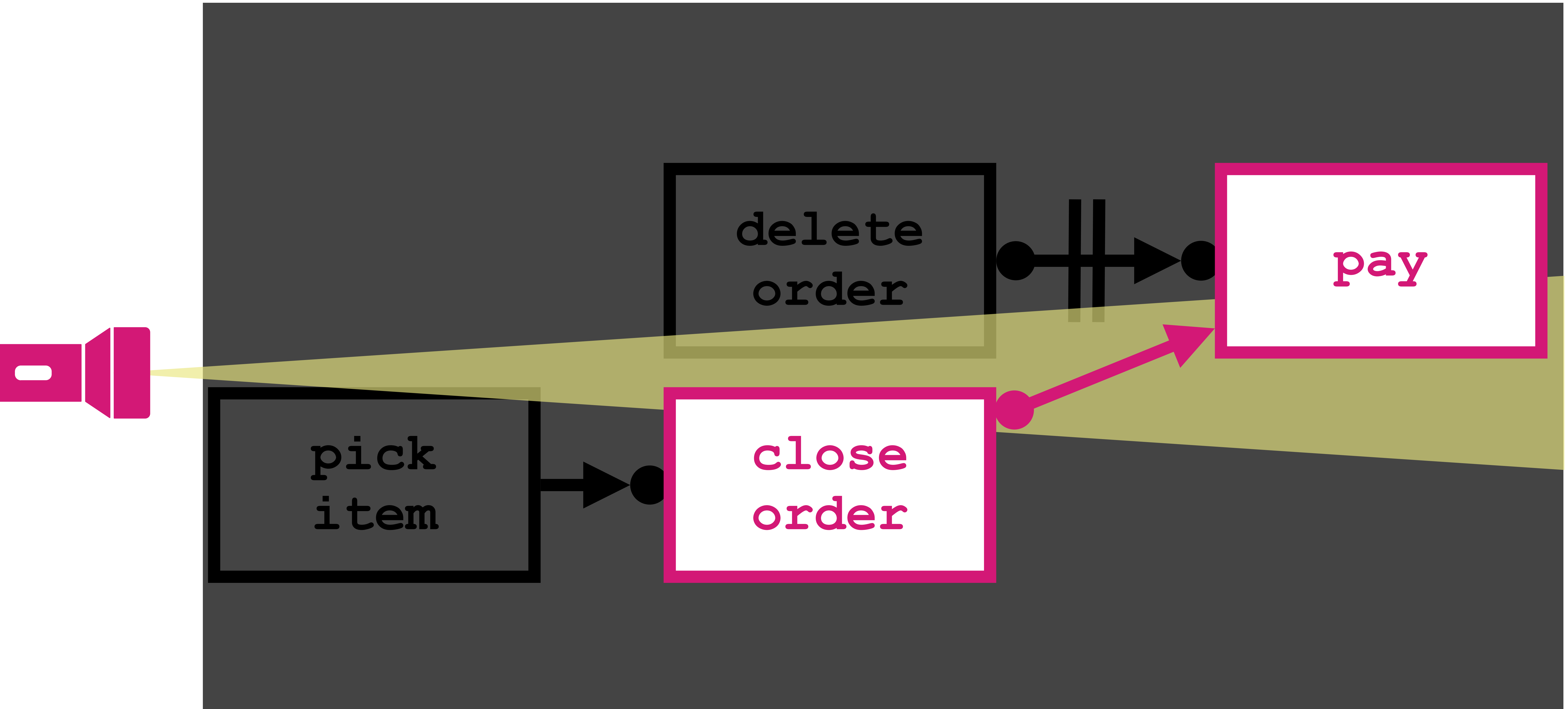
Semantics of Declare via LTLf

Each state contains a single activity



Atomic **propositions** are **activities**

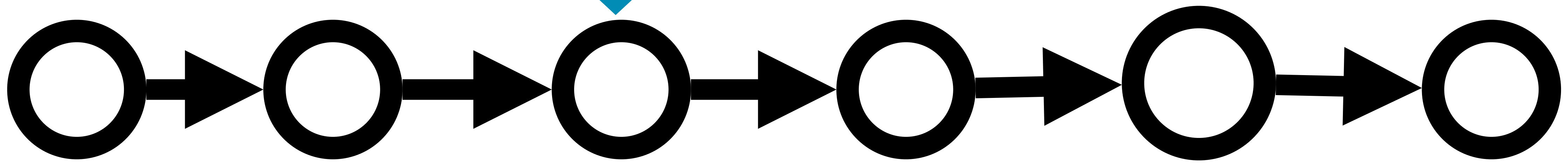
Each **constraint** is an **LTL formula** (built from template formulae)



$\Box(\mathtt{close} \rightarrow \Diamond \mathtt{pay})$

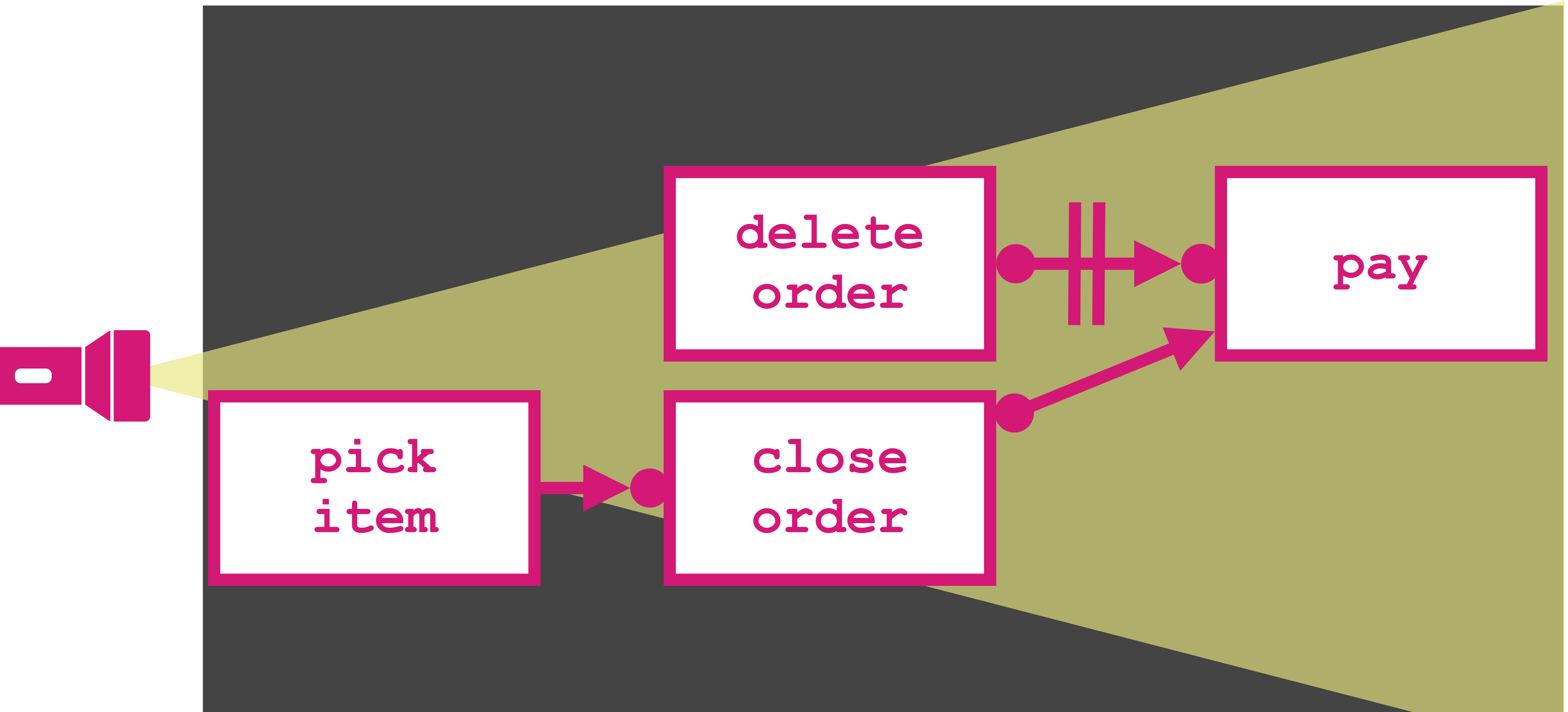
Semantics of Declare via LTLf

Each state contains a single activity



Atomic **propositions** are **activities**

A **Declare specification** is the **conjunction of its constraint formulae**



$$\Box(\mathtt{close} \rightarrow \Diamond\mathtt{pay})$$
$$\wedge\,\Box(\mathtt{close} \rightarrow \Diamondminus\mathtt{item})$$
$$\wedge\,\Box(\mathtt{cancel} \rightarrow \neg\Box\mathtt{pay})$$

An unconventional use of logics!

From …

Temporal logics for **specifying** (un)desired **properties of a dynamic system**

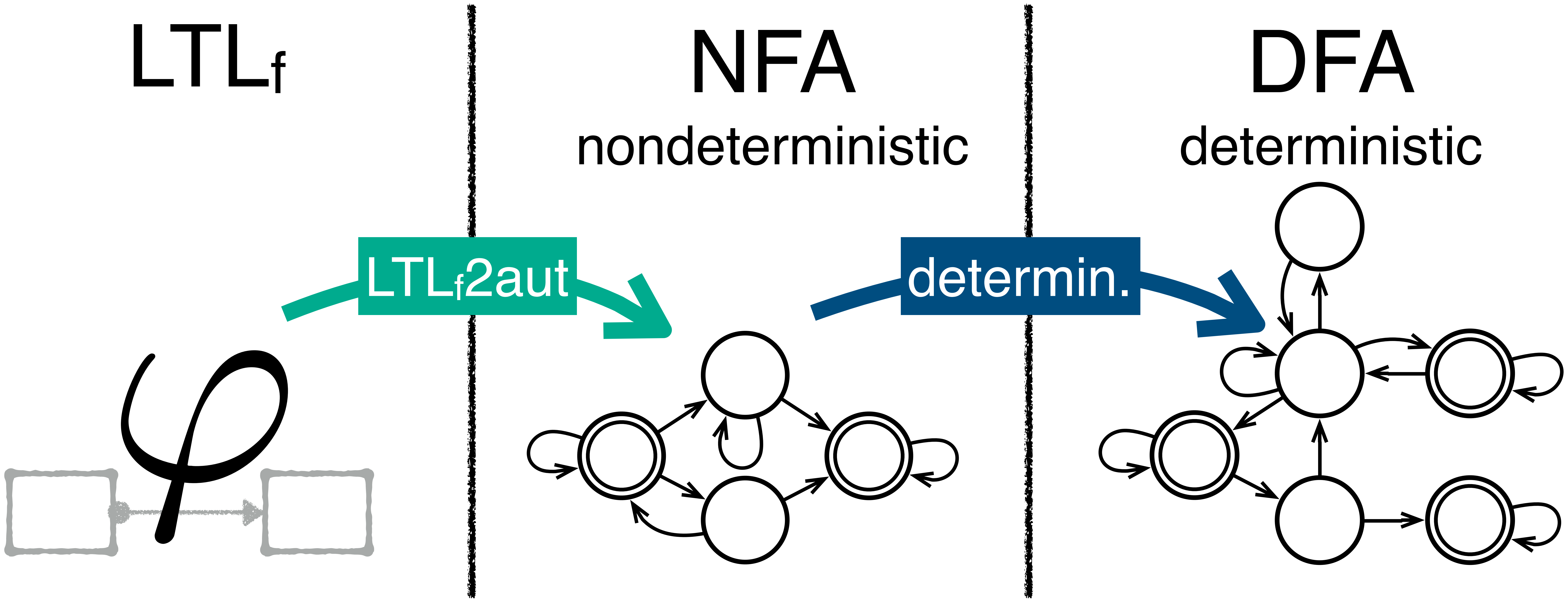
… to …

Temporal logics for **specifying the dynamic system itself**

From Declare to automata

Thanks to finite traces: good old finite-state automata!

$$t \models \varphi \text{ iff } t \in \mathcal{L}(\mathcal{A}_\varphi)$$



LTL$_f$

NFA

nondeterministic

DFA

deterministic

LTL$_f$2aut

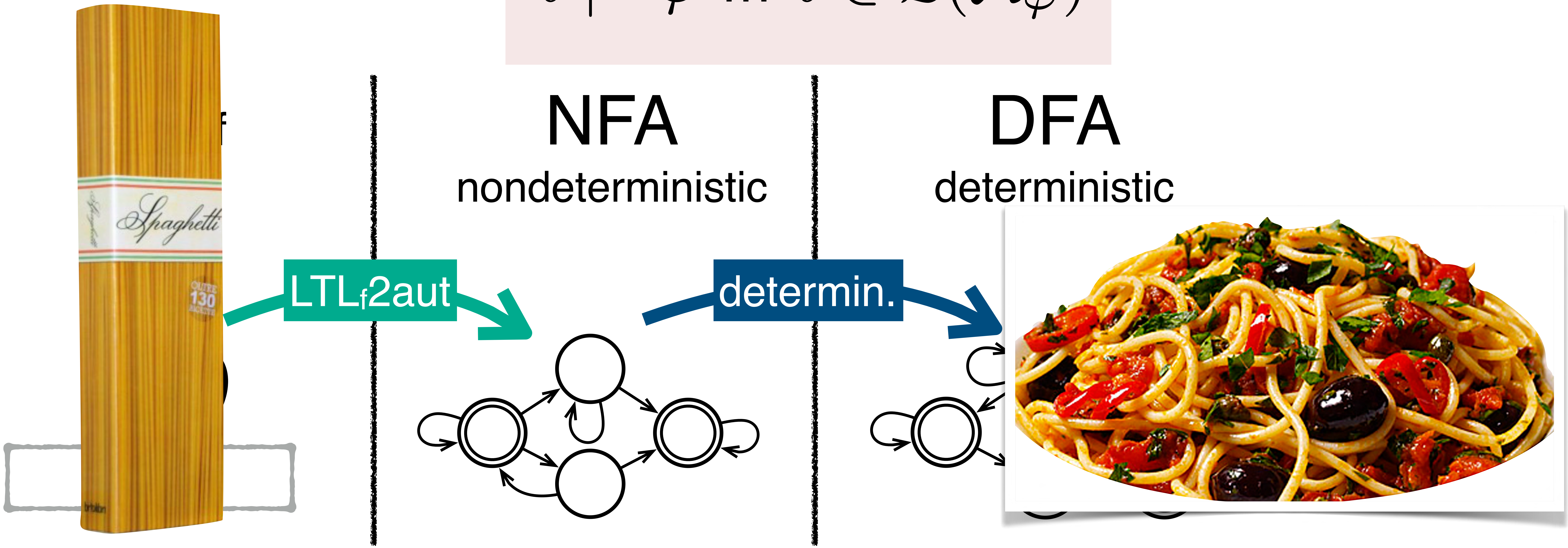
determin.

[DeGiacomoVardi,IJCAI2013]

[_____,TOSEM2022]

Vision realised!

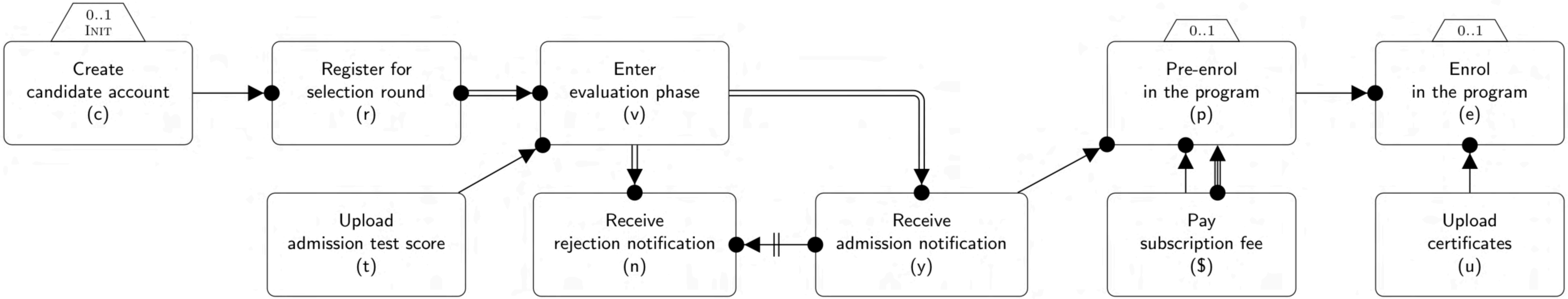
$$t \models \varphi \text{ iff } t \in \mathcal{L}(\mathcal{A}_\varphi)$$



[DeGiacomoVardi,IJCAI2013]
[_____,TOSEM2022]

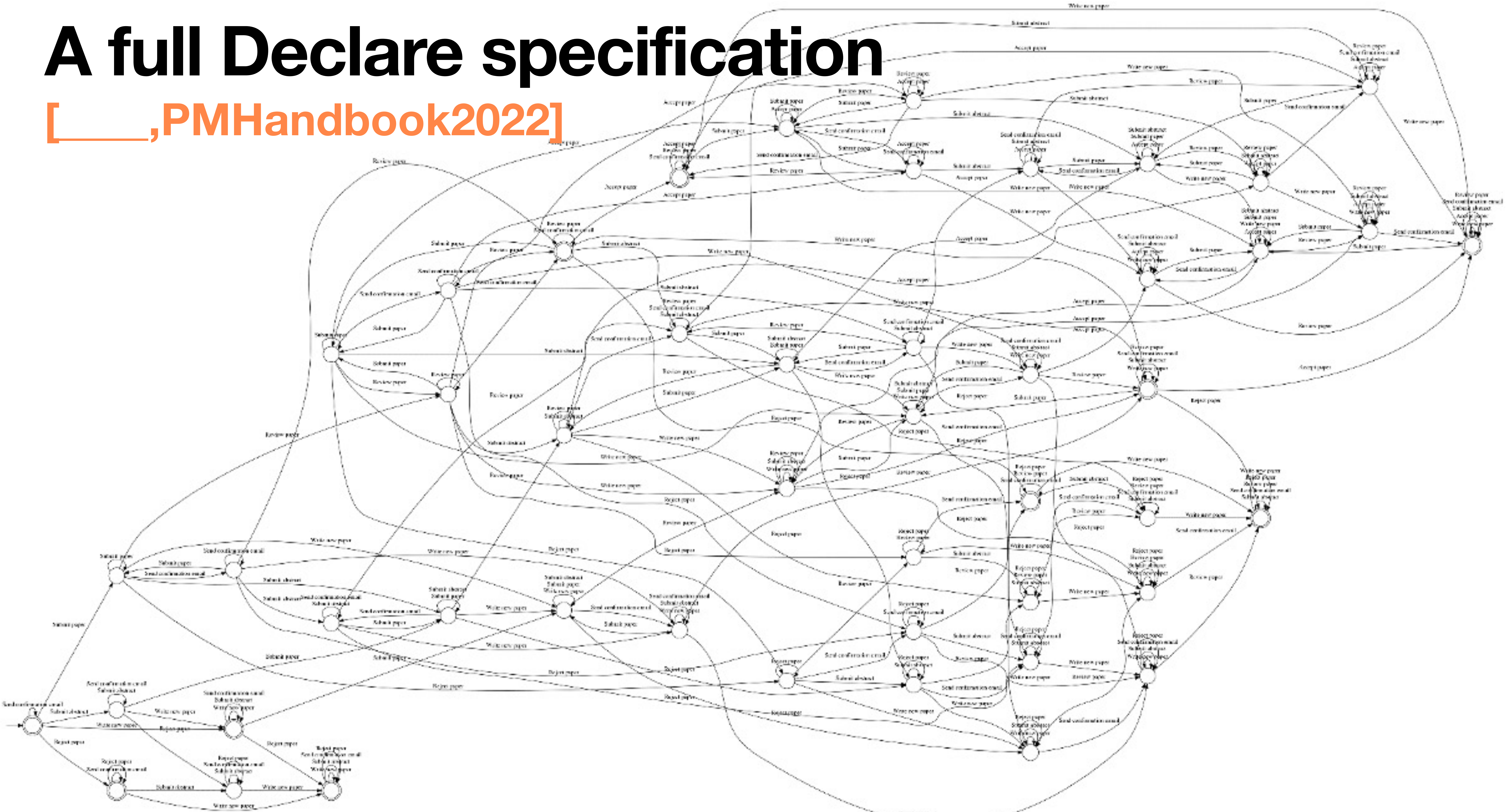
A full Declare specification

[____,PMHandbook2022]



A full Declare specification

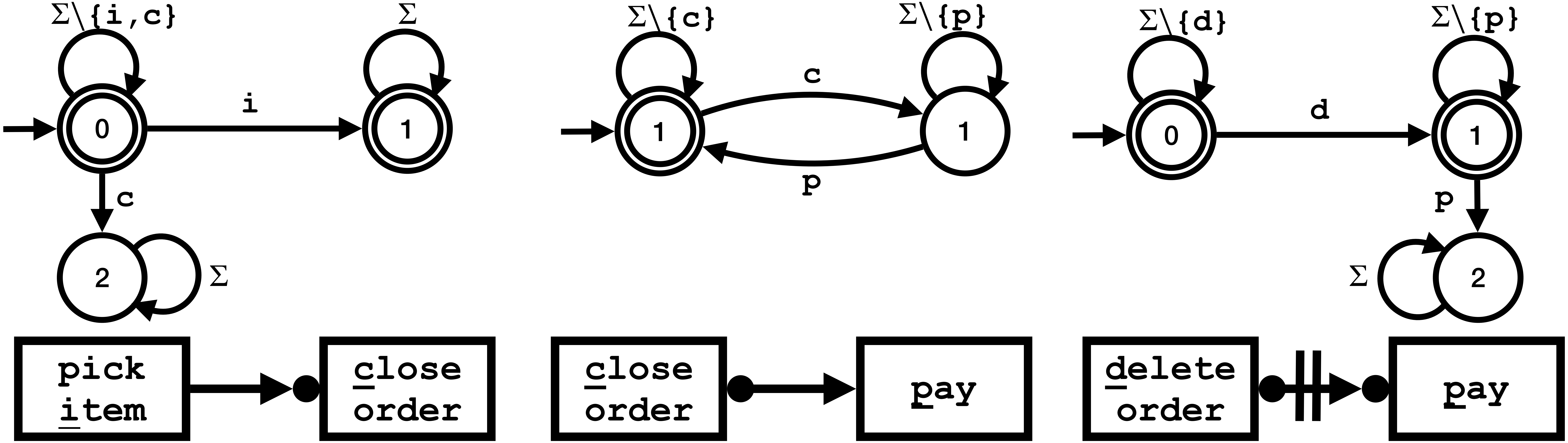
[____,PMHandbook2022]



Constraint automata

Template: pre-compiled into a DFA

Constraint: grounds the template DFA on specific activities



Combining constraints

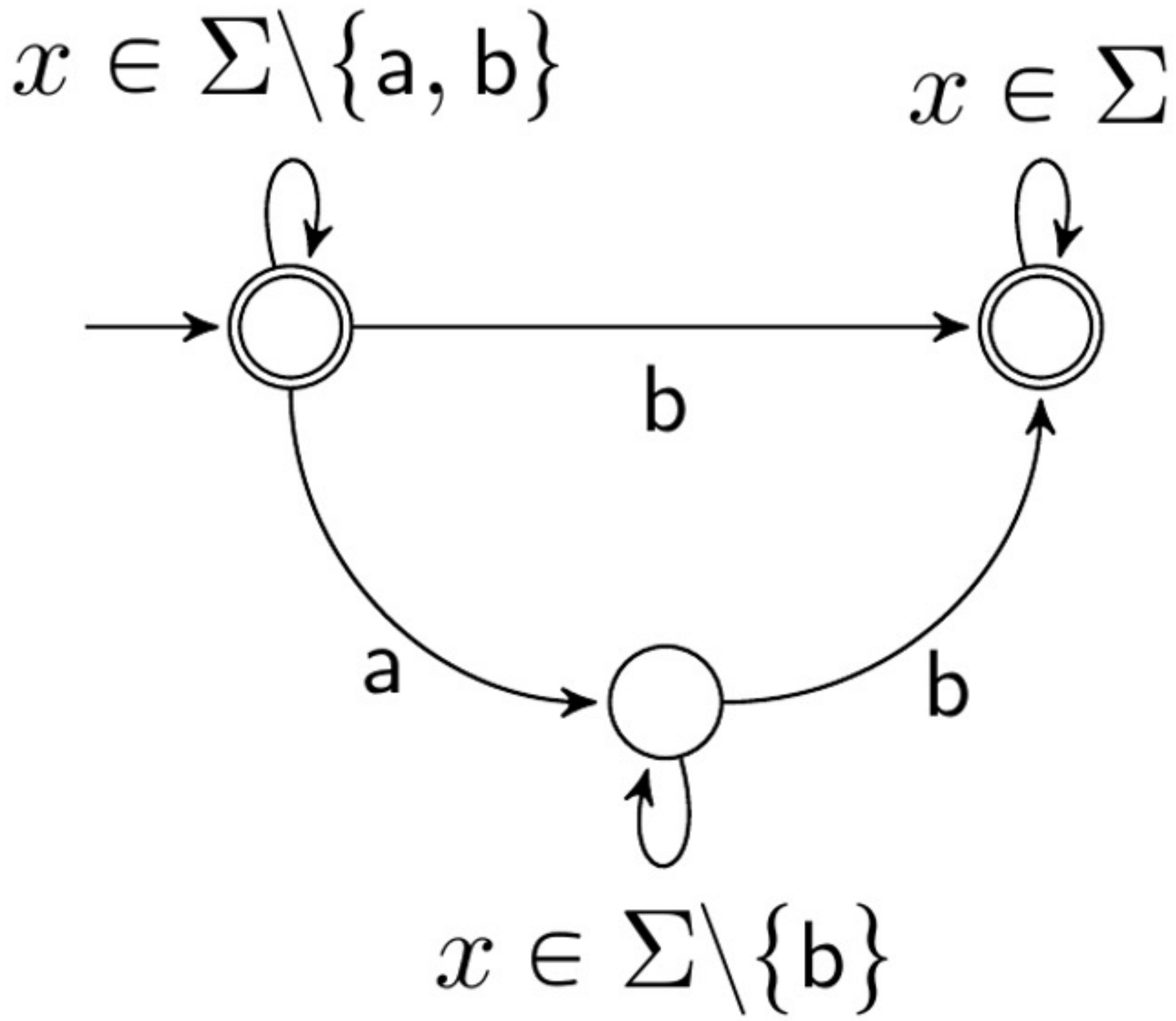
responded existence(a,b)

response(a,c)

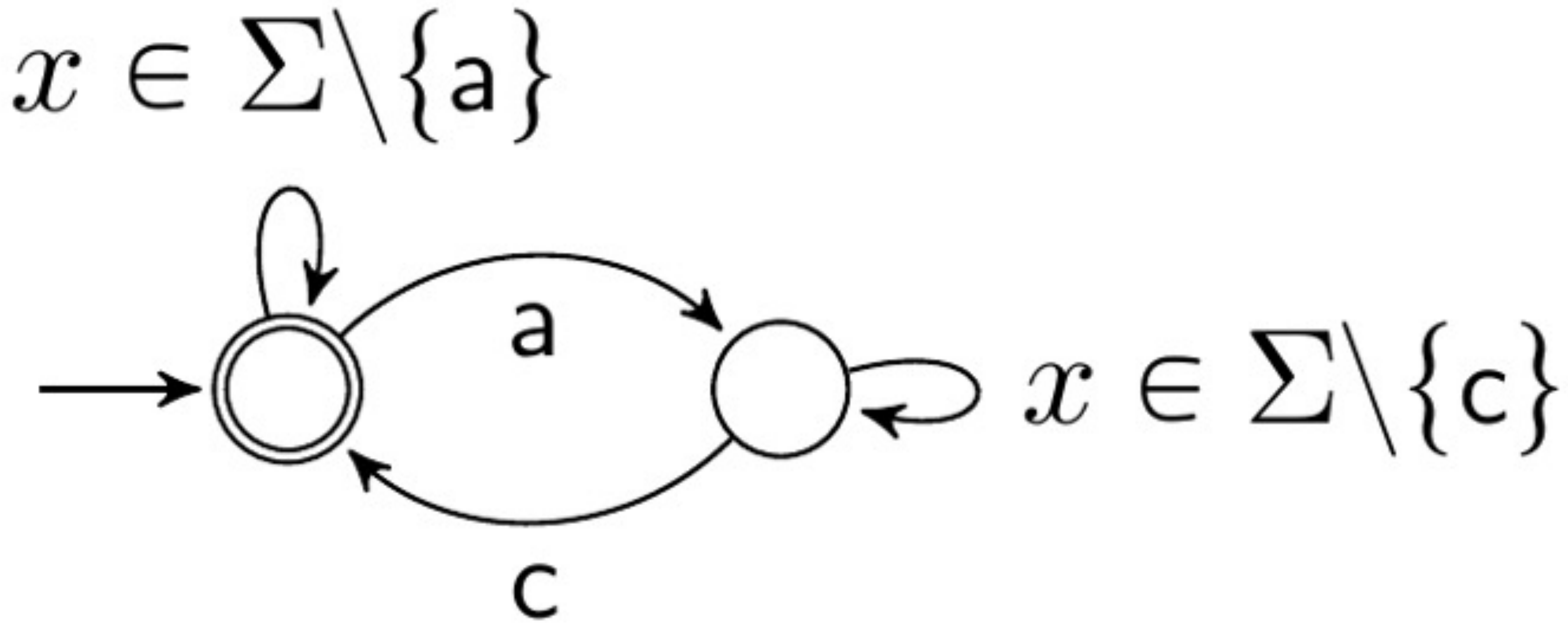
Combining constraints

responded existence(a,b)

response(a,c)



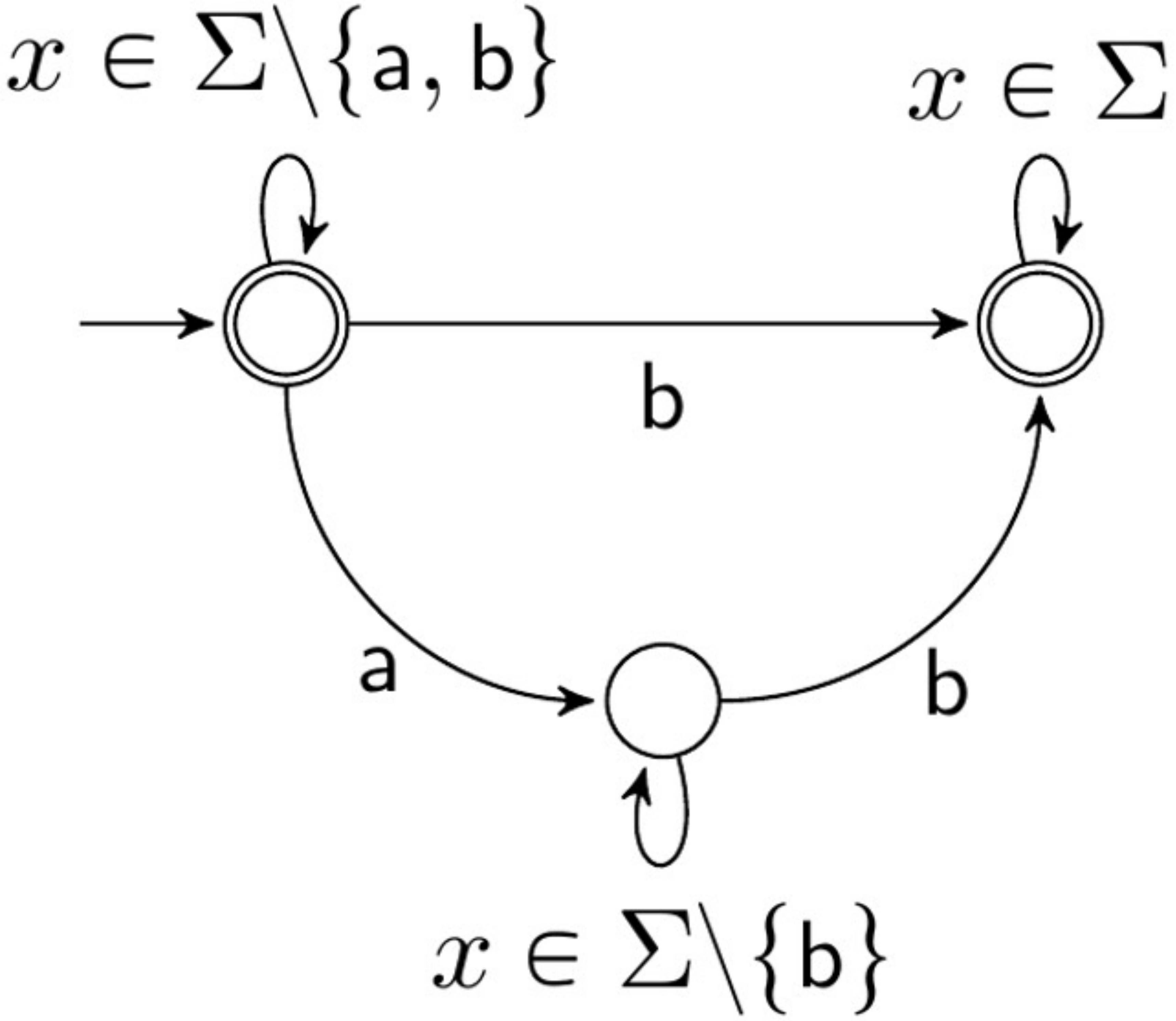
$x \in \Sigma \backslash \{a, b\}$
$x \in \Sigma$
b
a
b
$x \in \Sigma \backslash \{b\}$

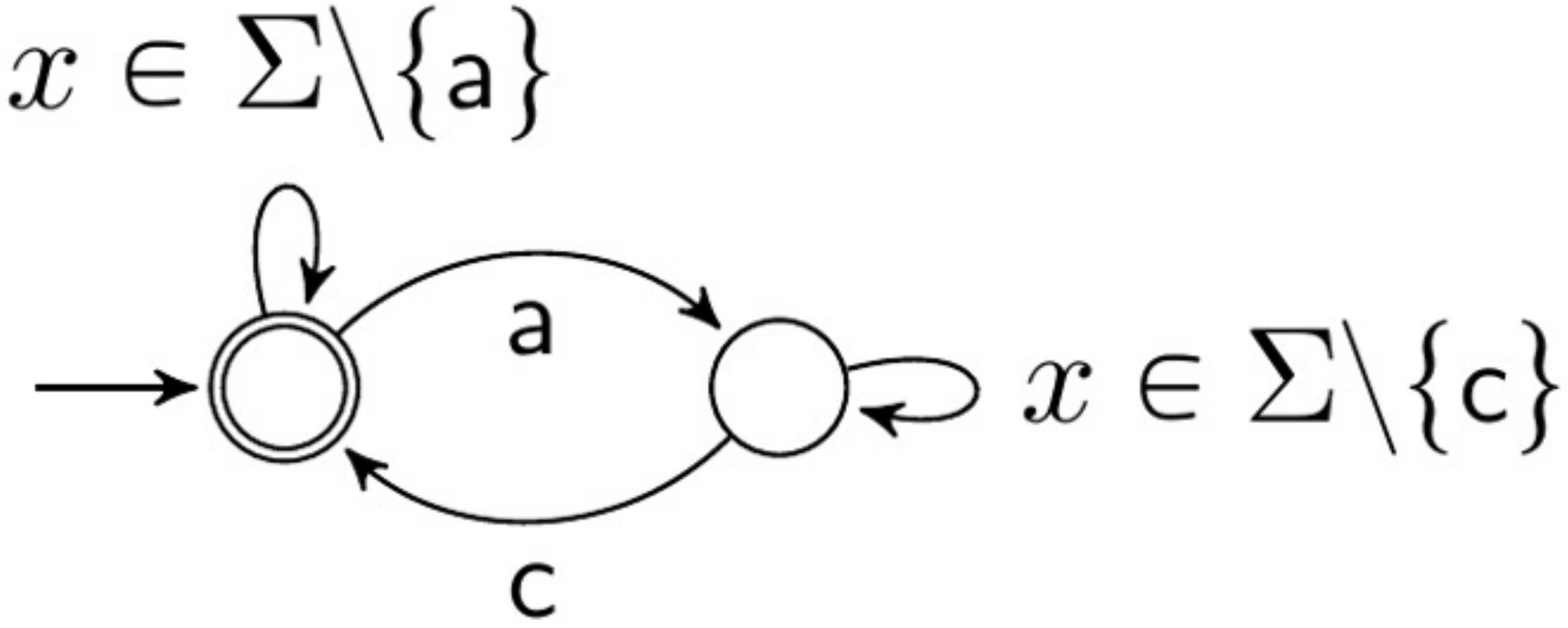


$x \in \Sigma \backslash \{a\}$
a
$x \in \Sigma \backslash \{c\}$
c

Combining constraints

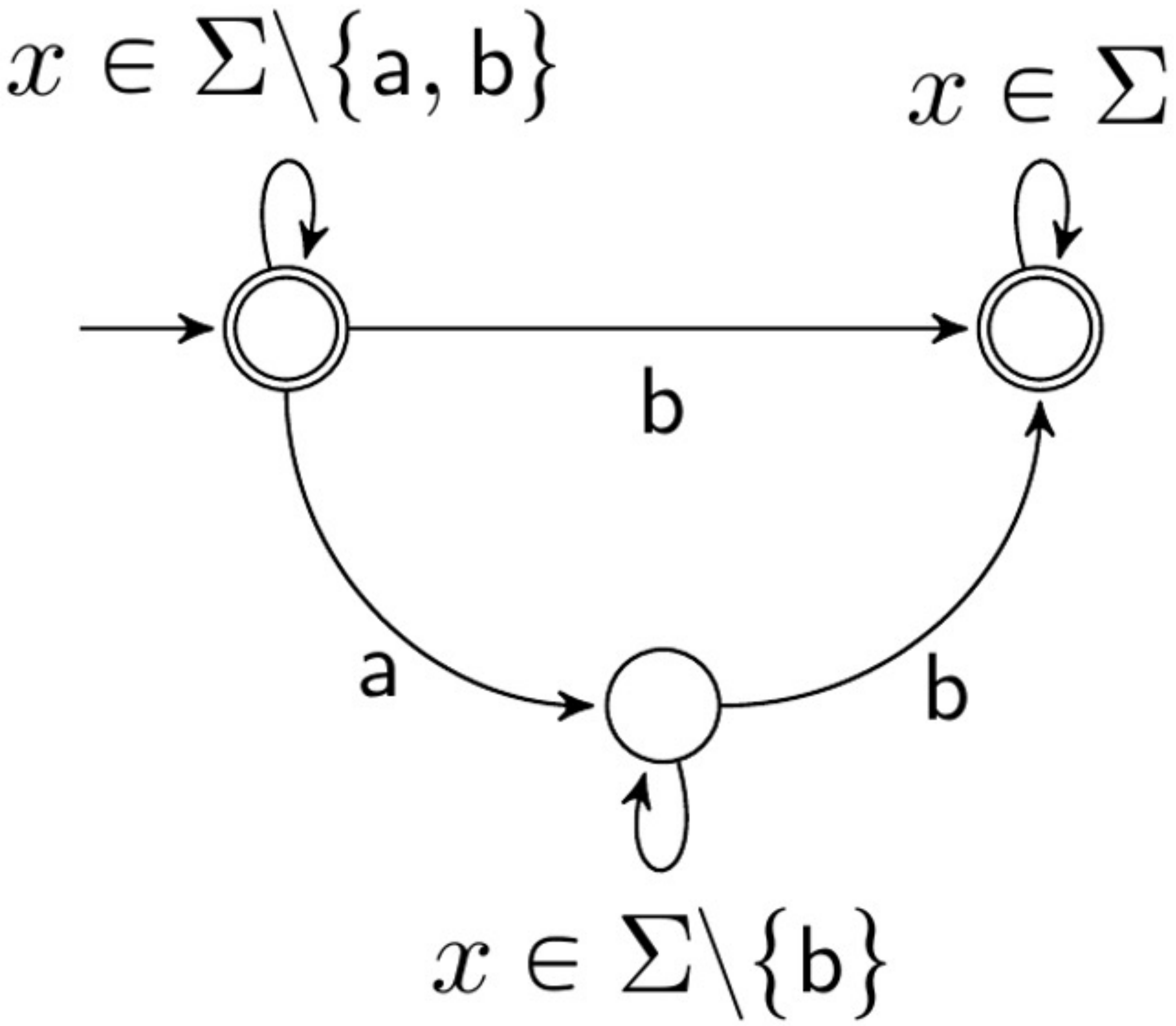
responded existence(a,b) AND **response(a,c)**


$x \in \Sigma \backslash \{a, b\}$
$x \in \Sigma$
b
a
b
$x \in \Sigma \backslash \{b\}$


$x \in \Sigma \backslash \{a\}$
a
$x \in \Sigma \backslash \{c\}$
c

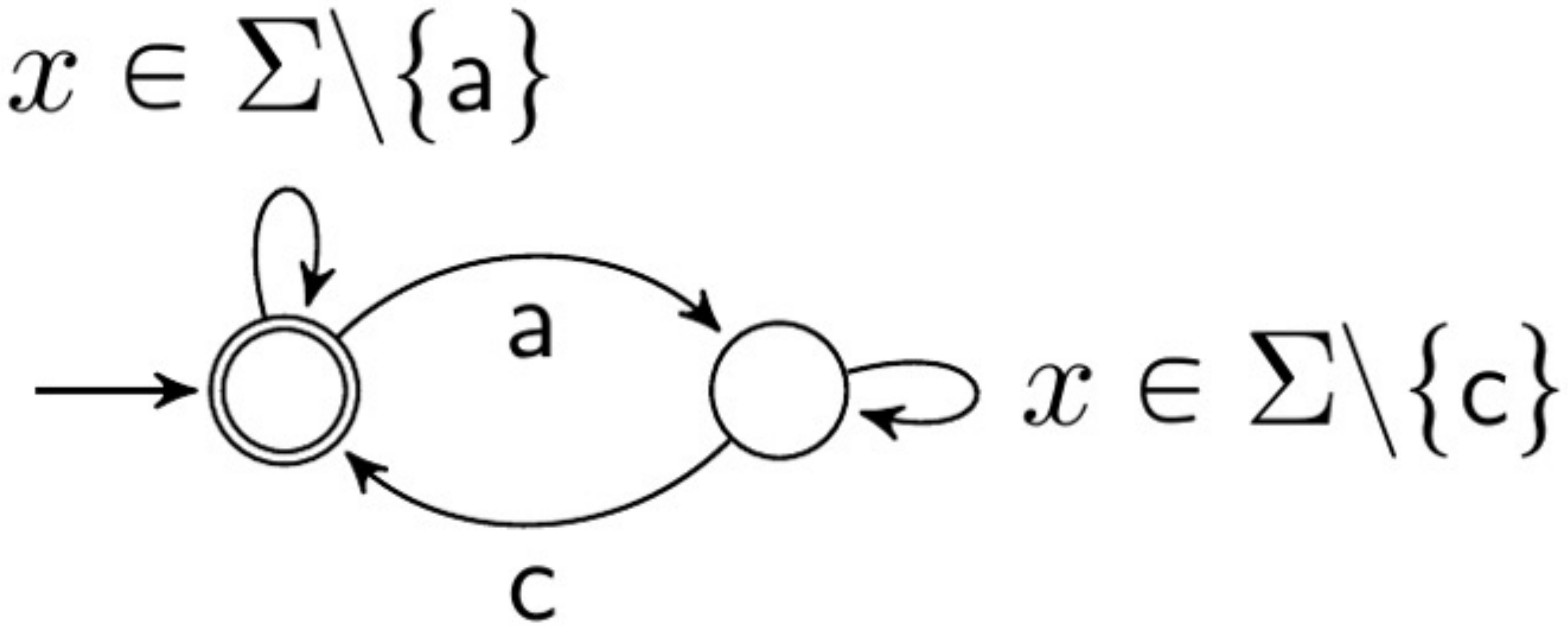
Combining constraints

responded existence(a,b) AND **response(a,c)**



$x \in \Sigma \backslash \{a, b\}$
$x \in \Sigma$
b
a
b
$x \in \Sigma \backslash \{b\}$

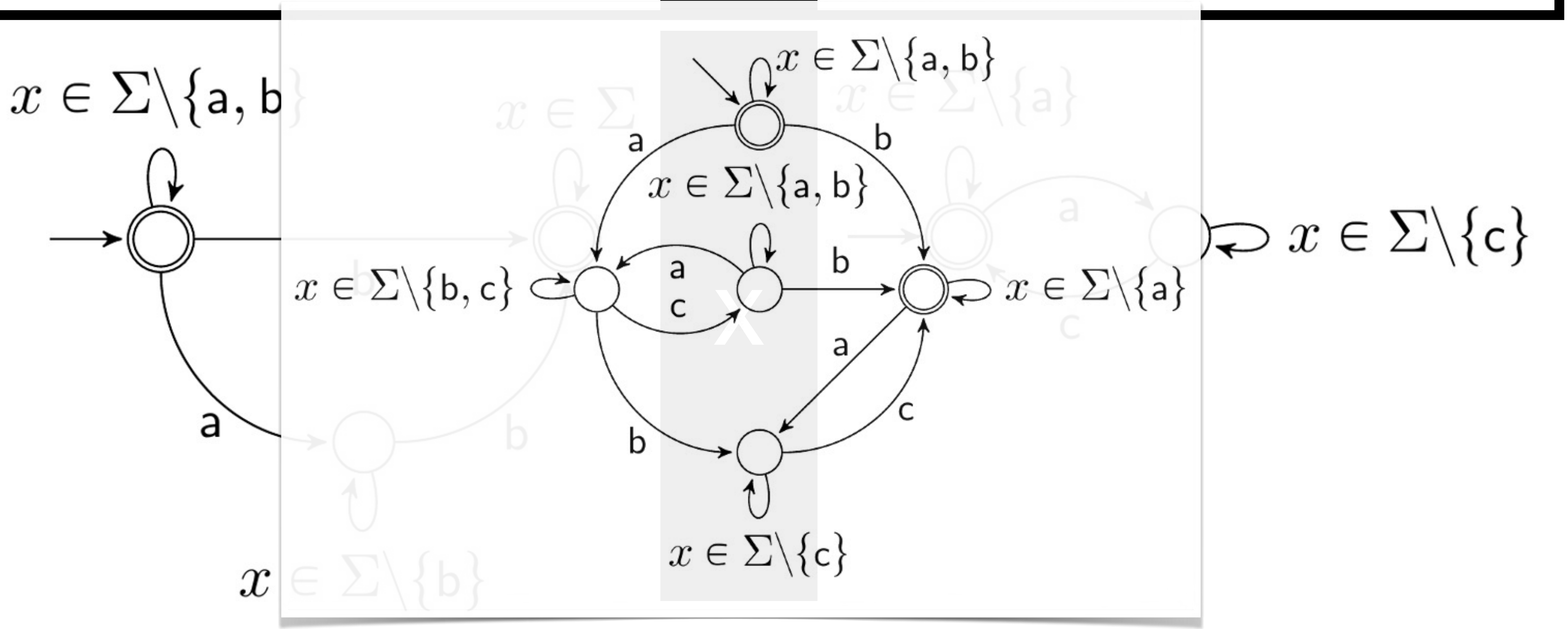
X



$x \in \Sigma \backslash \{a\}$
a
$x \in \Sigma \backslash \{c\}$
c

Combining constraints

responded existence(a,b) AND **response(a,c)**



$x \in \Sigma\backslash\{a, b\}$
a
b
$x \in \Sigma\backslash\{a, b\}$
$x \in \Sigma\backslash\{a, b\}$
$x \in \Sigma$
$x \in \Sigma\backslash\{a\}$
a
$x \in \Sigma\backslash\{b, c\}$
a
c
b
X
$x \in \Sigma\backslash\{a\}$
$x \in \Sigma\backslash\{c\}$
c
a
c
a
b
b
$x \in \Sigma\backslash\{c\}$
$x \in \Sigma\backslash\{b\}$

From local automata to global automaton

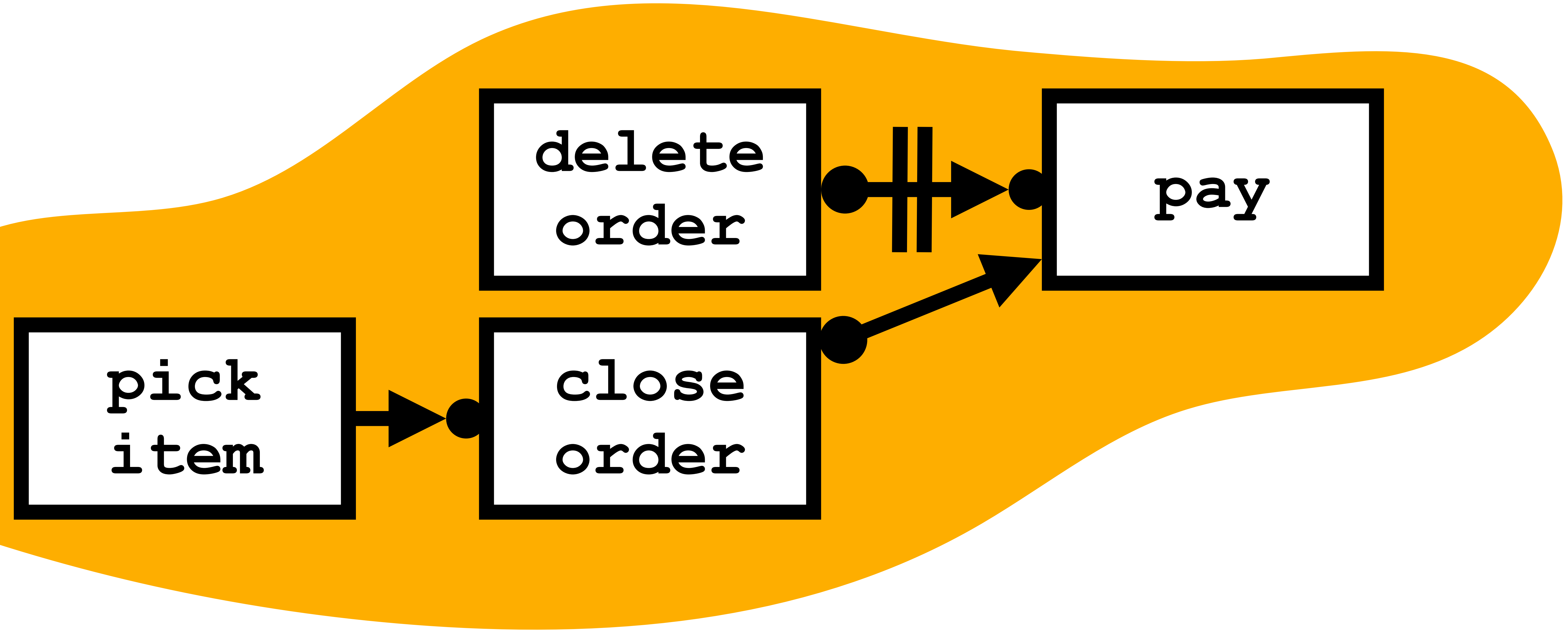
Entire specification: **product automaton** of all local automata

- Corresponds to the **automaton of the conjunction** of all formulae
- Many **optimisations available**

Declare **specification consistent** if and only if its **global automaton is non-empty**

Framework in action: enactment and monitoring

The global automaton is an execution engine

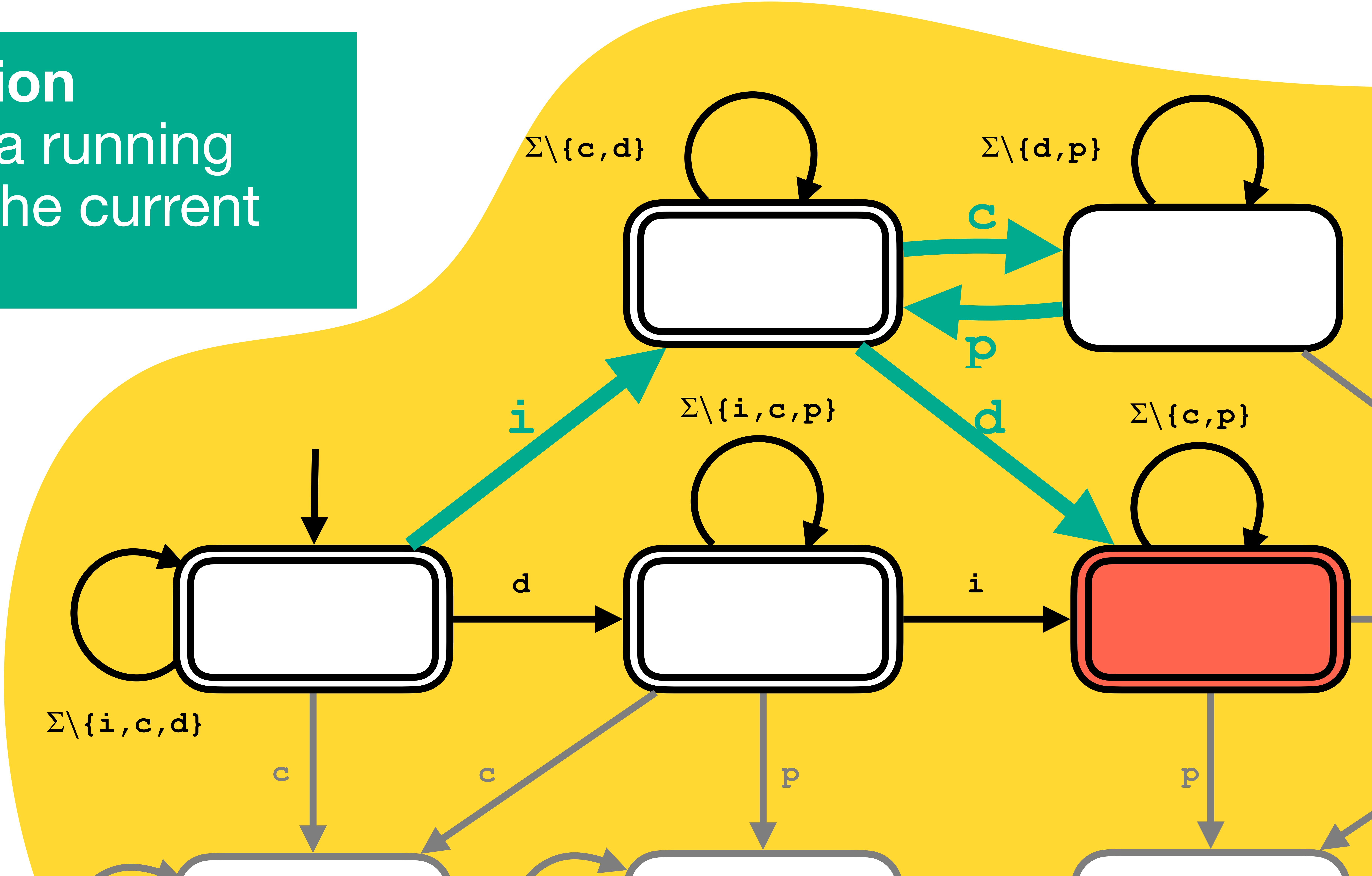


The global automaton is an execution engine

delete order
pay
pick item
close order
Σ\{c,d}
Σ\{d,p}
c
p
i
Σ\{i,c,p}
d
Σ\{c,p}
d
i
Σ\{i,c,d}
c
c
p
p

The global automaton is an execution engine

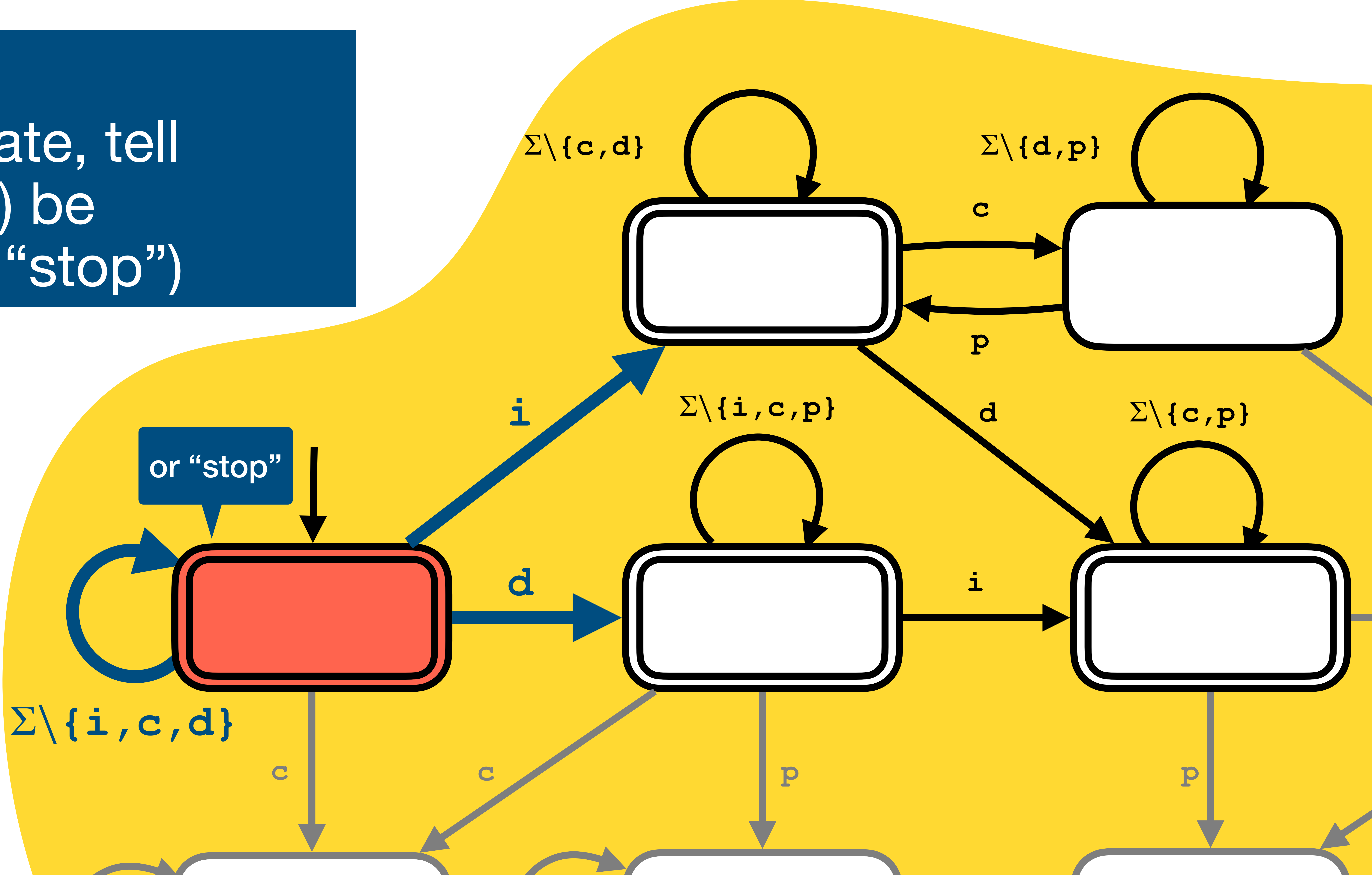
1. History recognition
given the history of a running instance, compute the current state (or reject)



The global automaton is an execution engine

2. Todo list
Given the current state, tell which tasks can(not) be executed next (also "stop")

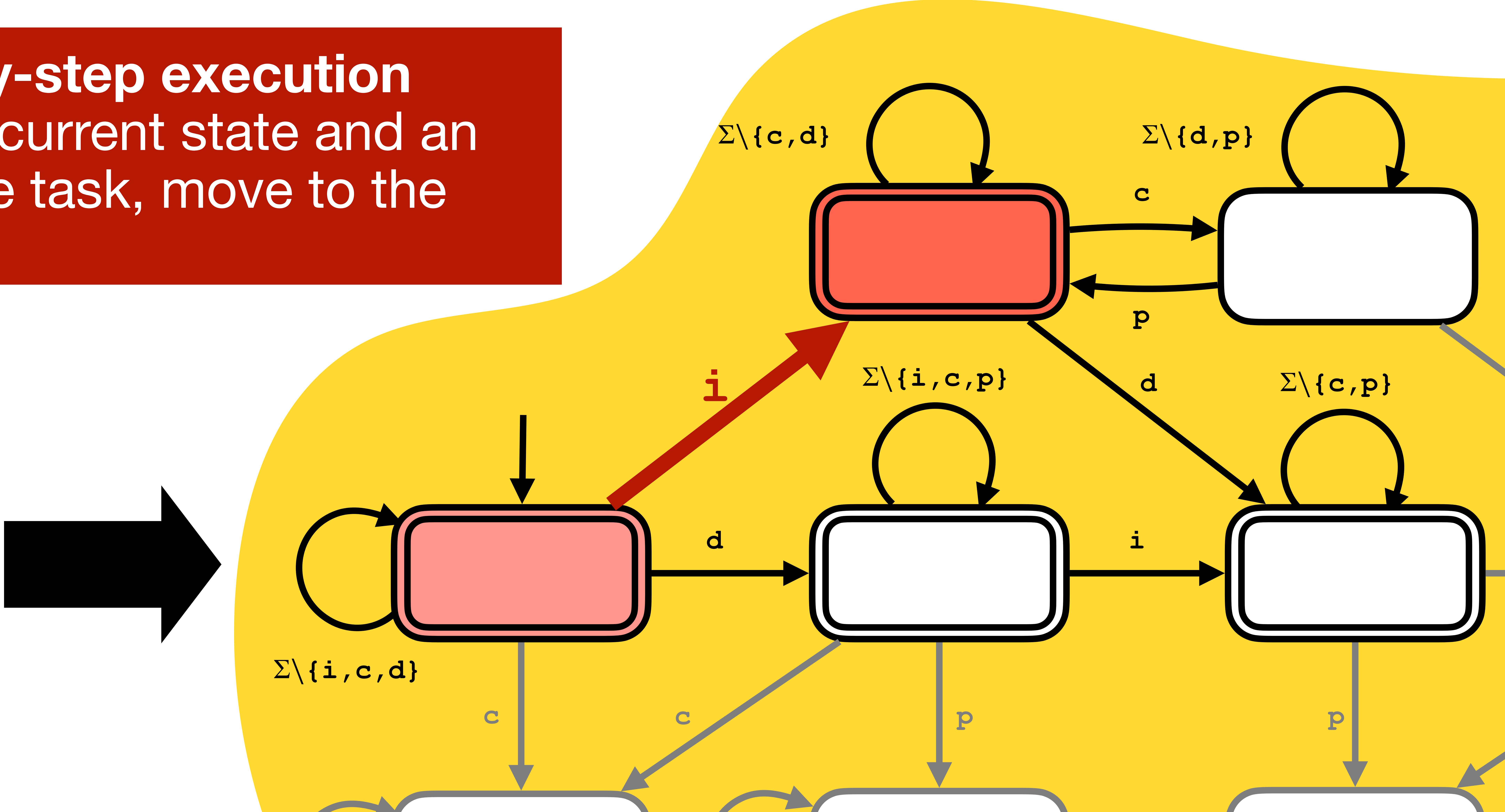
what to do?



The global automaton is an execution engine

3. Step-by-step execution
Given the current state and an executable task, move to the next state

do "i"

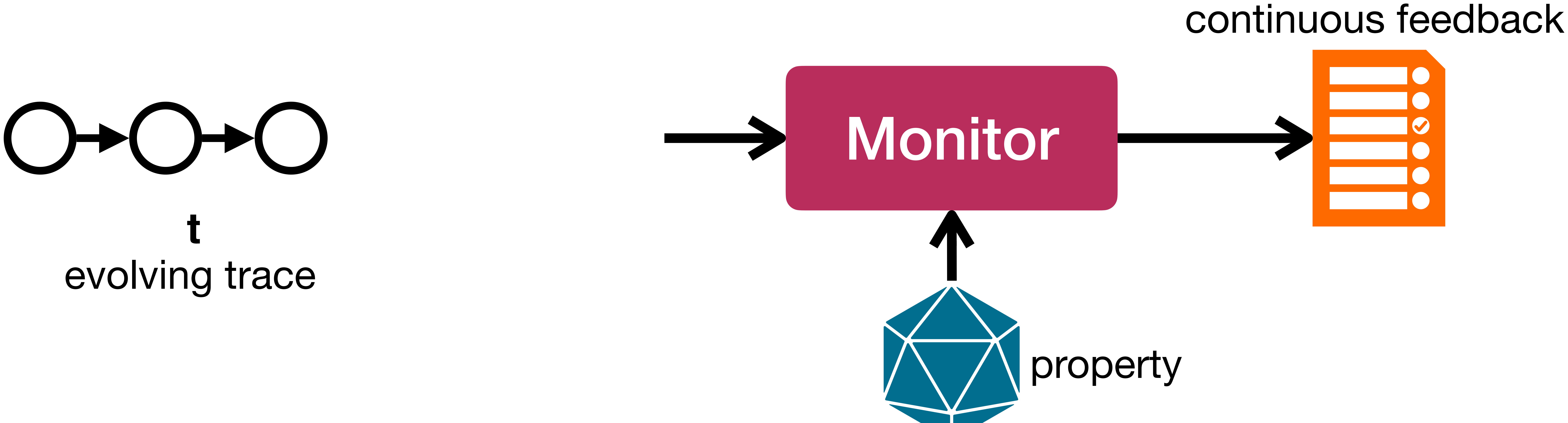


monitoring

Track a **running process execution** to **check conformance** with **properties** of interest

- Goal: Detect and report **fine-grained feedback** and **deviations**

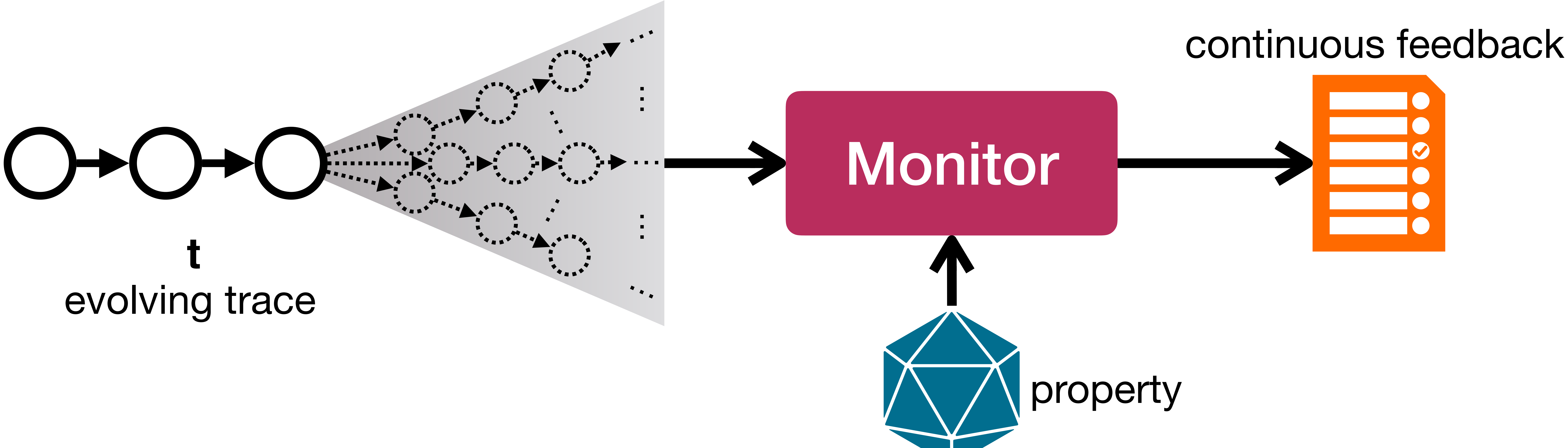
- Complementary to **predictive monitoring!**



(Anticipatory) monitoring

Track a **running process execution** to **check conformance** with **properties** of interest

- Goal: Detect and report **fine-grained feedback** and **deviations** **also considering the possible future continuations**
- Complementary to **predictive monitoring!**



Fine-grained feedback

As hard as satisfiability and validity!

Refined analysis of the “truth value” of a property

- looking into (all) **possible futures**

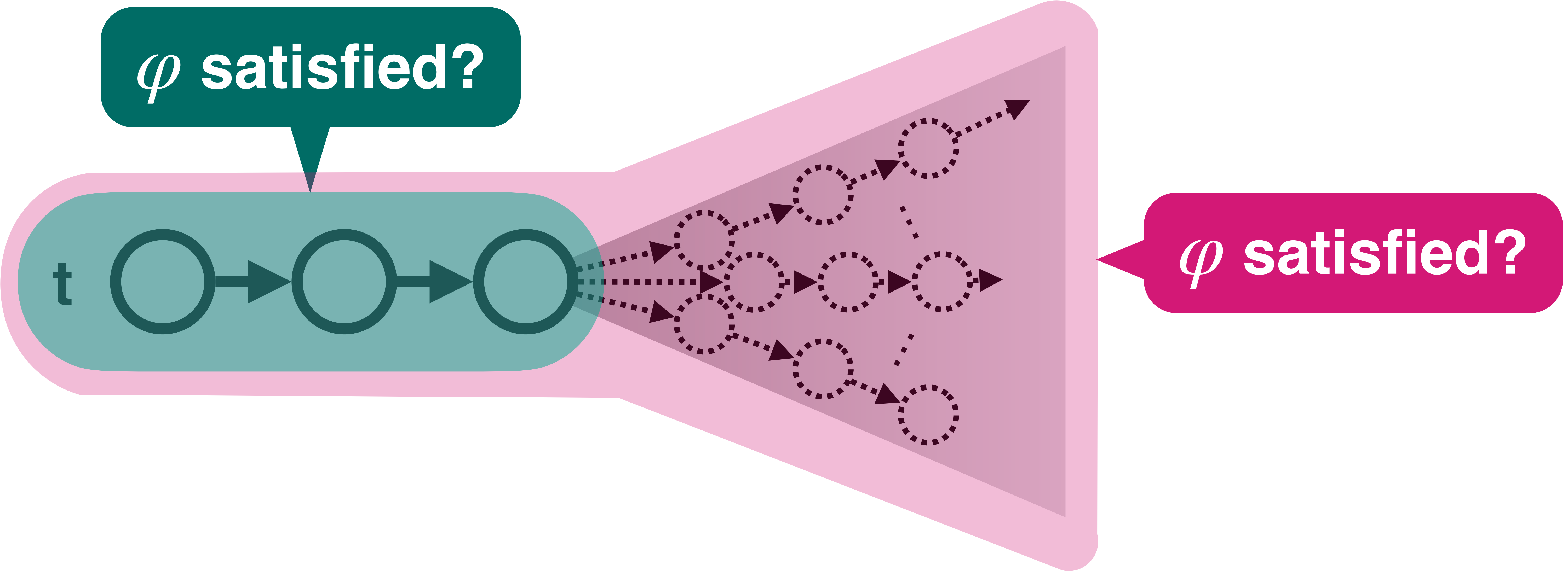
Fine-grained feedback

As hard as satisfiability and validity!

Refined analysis of the “truth value” of a property

- looking into (all) **possible futures**

Consider a partial trace **t**, and an LTLf formula φ…

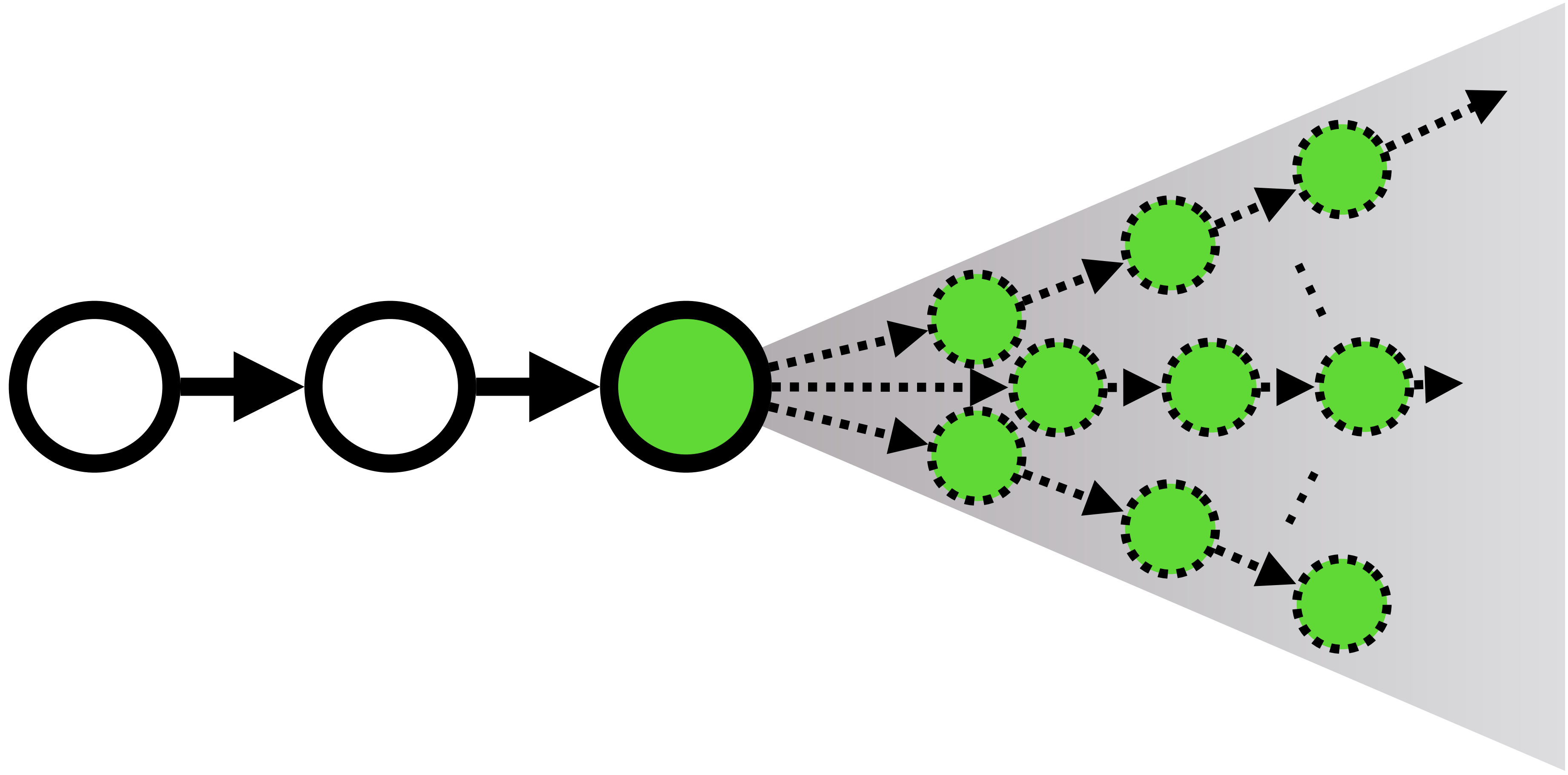


RV-LTL(f) truth values

[BauerEtAl,InfCom2010] [___,BPM2011] [___,TOSEM2023]

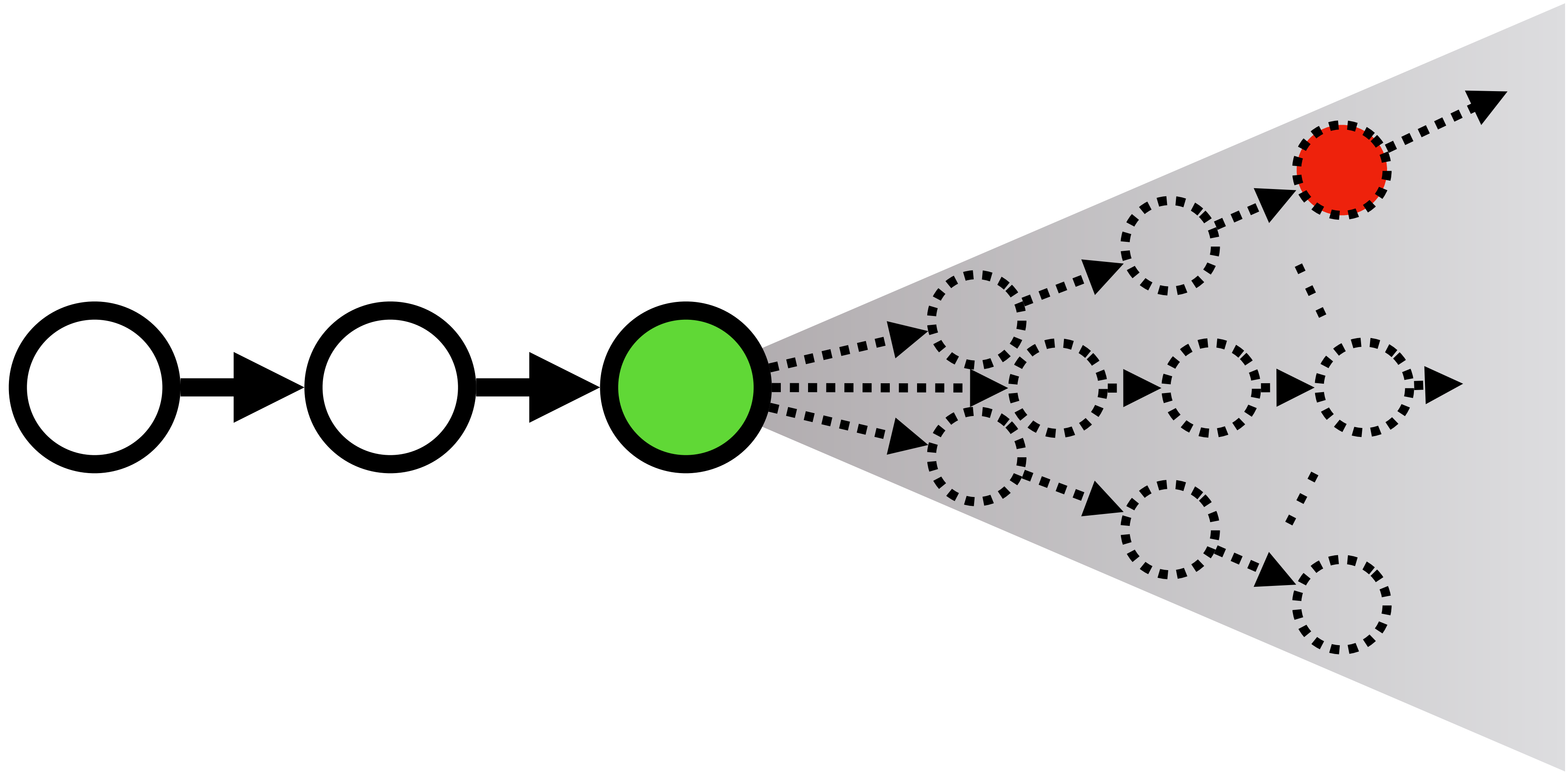
φ **permanently satisfied** by **t**

- **t** satisfies φ
- no matter how **t** continues, φ stays satisfied



φ **currently satisfied** by **t**

- **t** satisfies φ
- there is a continuation of **t** that violates φ

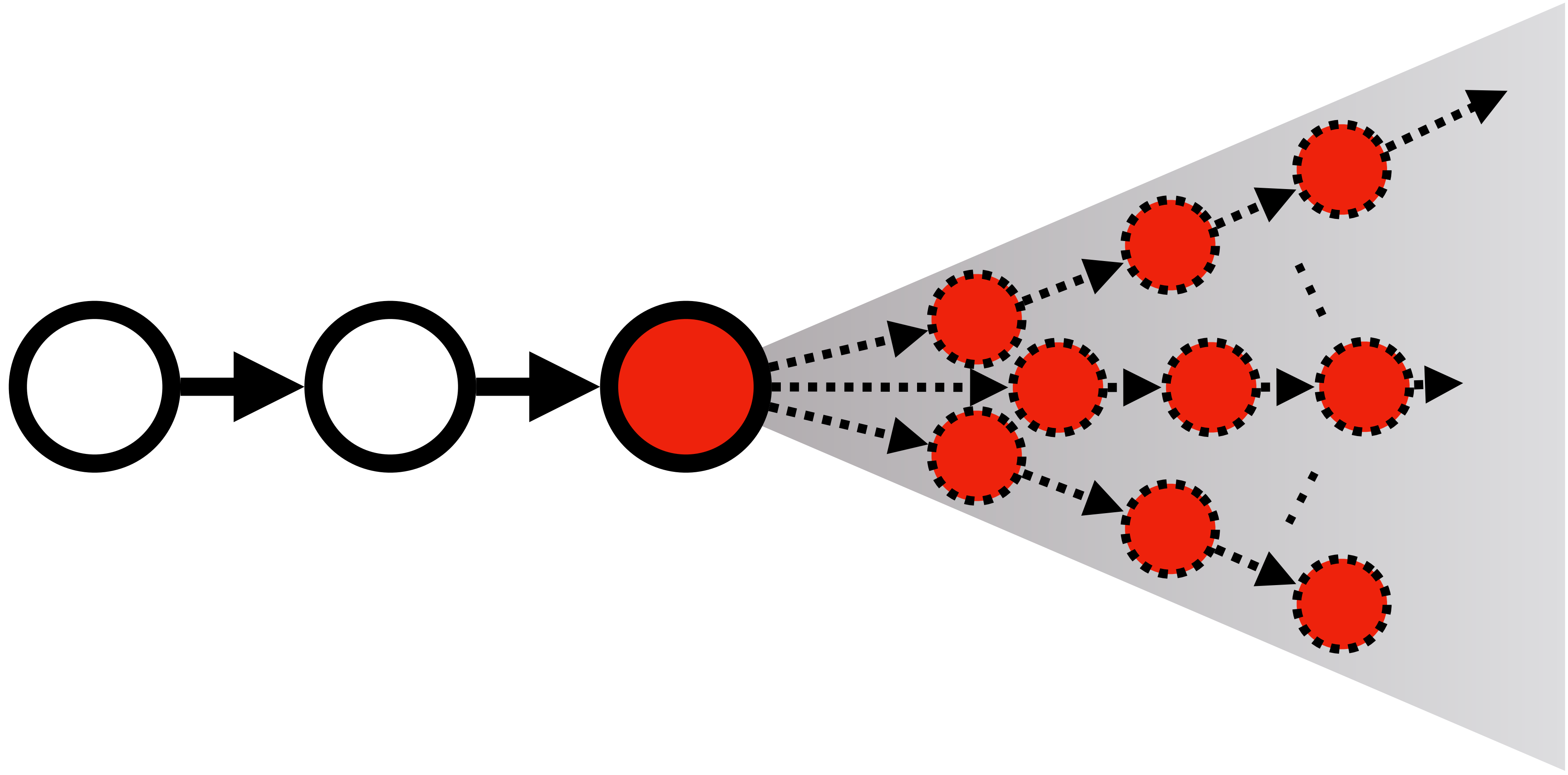


RV-LTL(f) truth values

[BauerEtAl,InfCom2010] [___,BPM2011] [___,TOSEM2023]

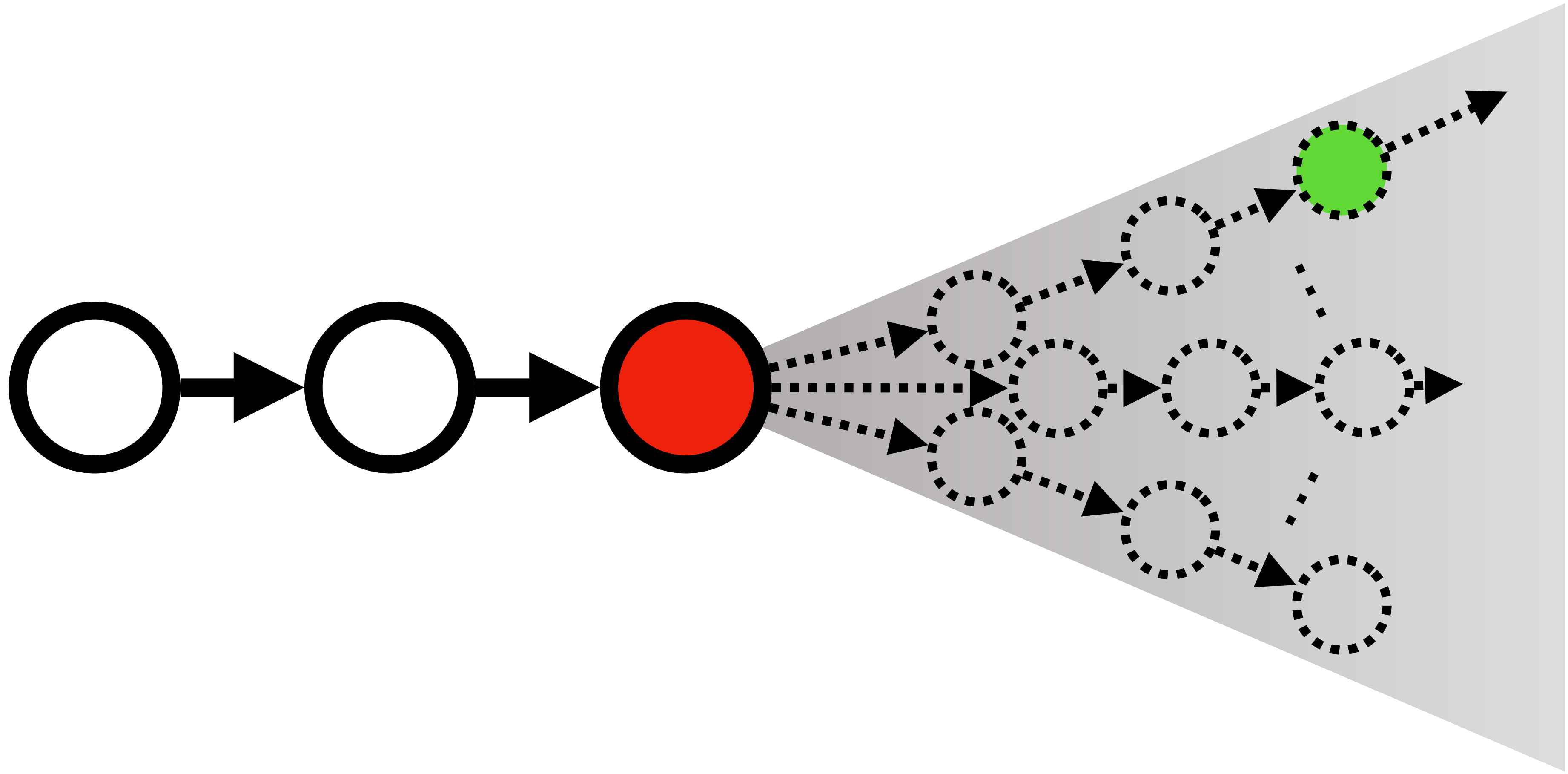
φ **permanently violated** by **t**

- **t** violates φ
- no matter how **t** continues, φ stays violated



φ **currently violated** by **t**

- **t** violates φ
- there is a continuation of **t** that satisfies φ

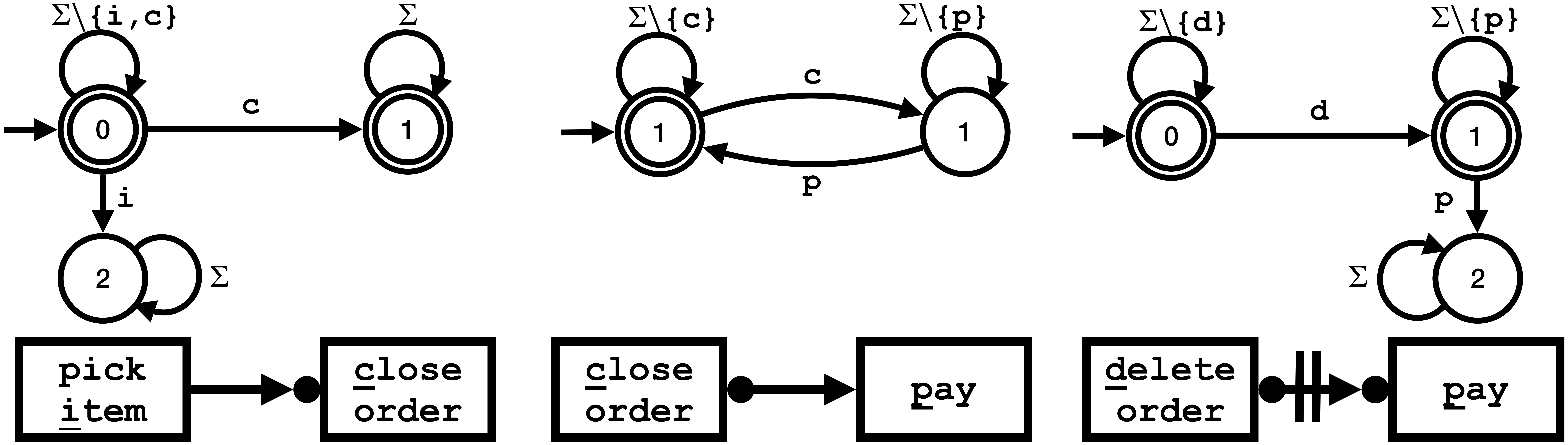


RV-LTL on finite traces

[____,BPM2011] [____,BPM2014] [____,TOSEM2022]

Suffixes of the current trace: each with **unbounded, finite length**

- RV-LTL truth values **encoded as regular expressions** -> **all formulae of LTLf are monitorable**



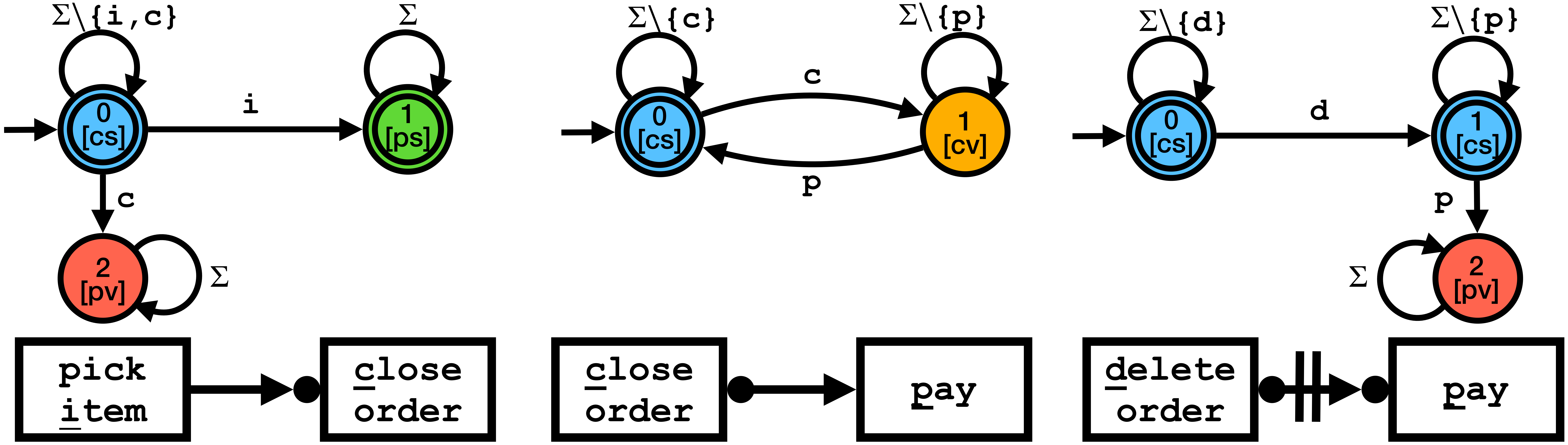
RV-LTL on finite traces

[____,BPM2011] [____,BPM2014] [____,TOSEM2022]

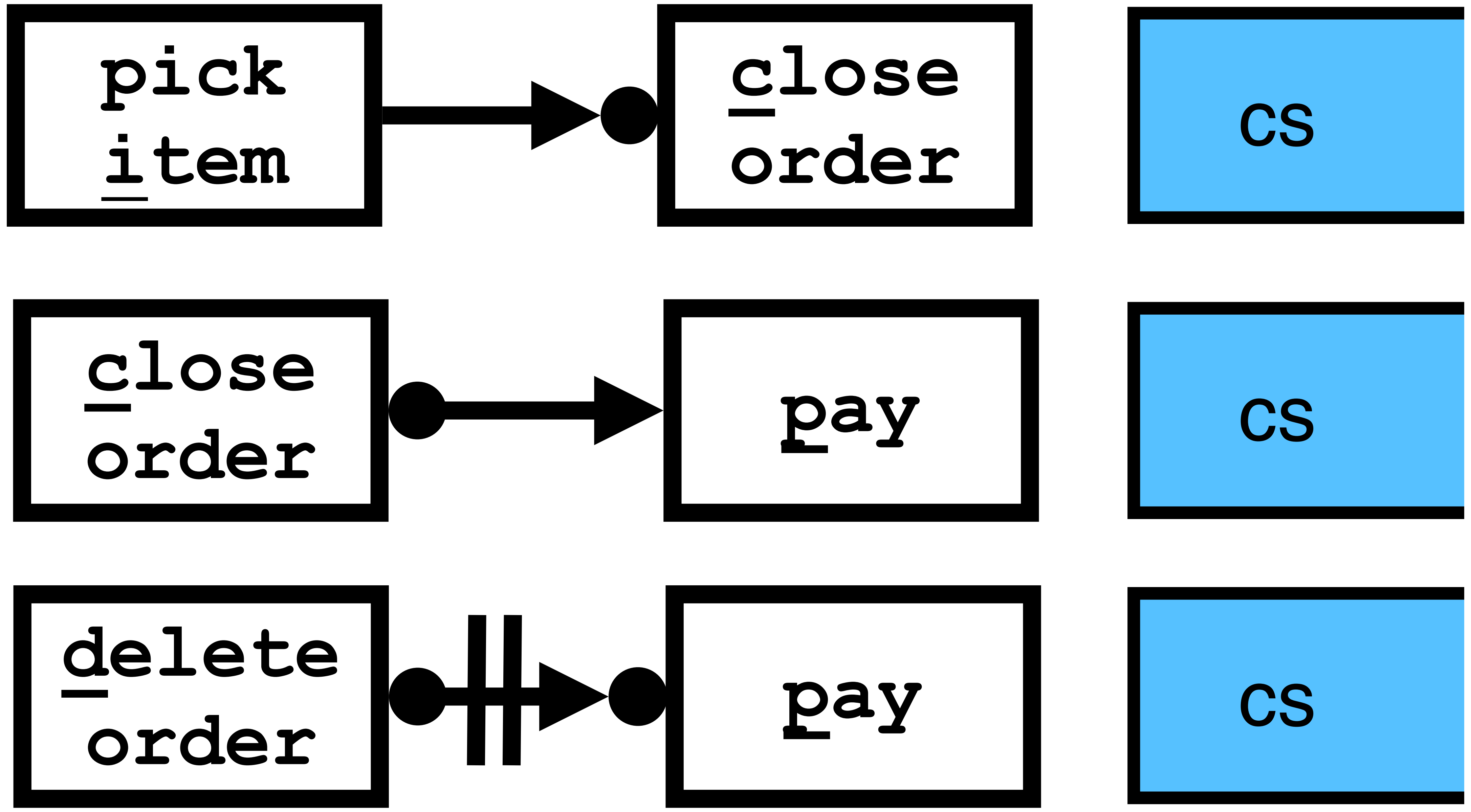
Suffixes of the current trace: each with **unbounded, finite length**

- RV-LTL truth values **encoded as regular expressions** -> **all formulae of LTLf are monitorable**

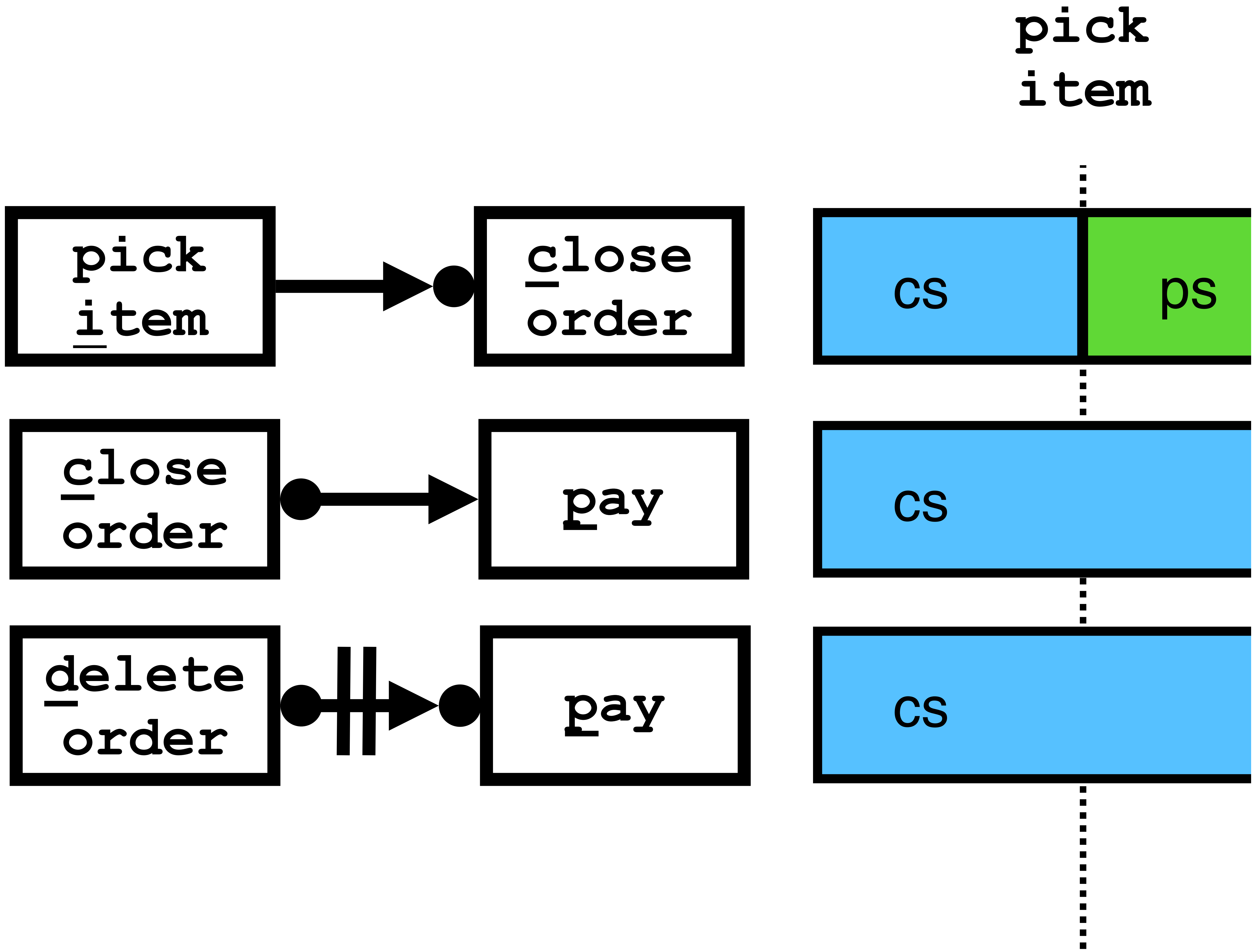
Operationally: **color** each DFA state with an **RV-LTL value** via simple reachability checks



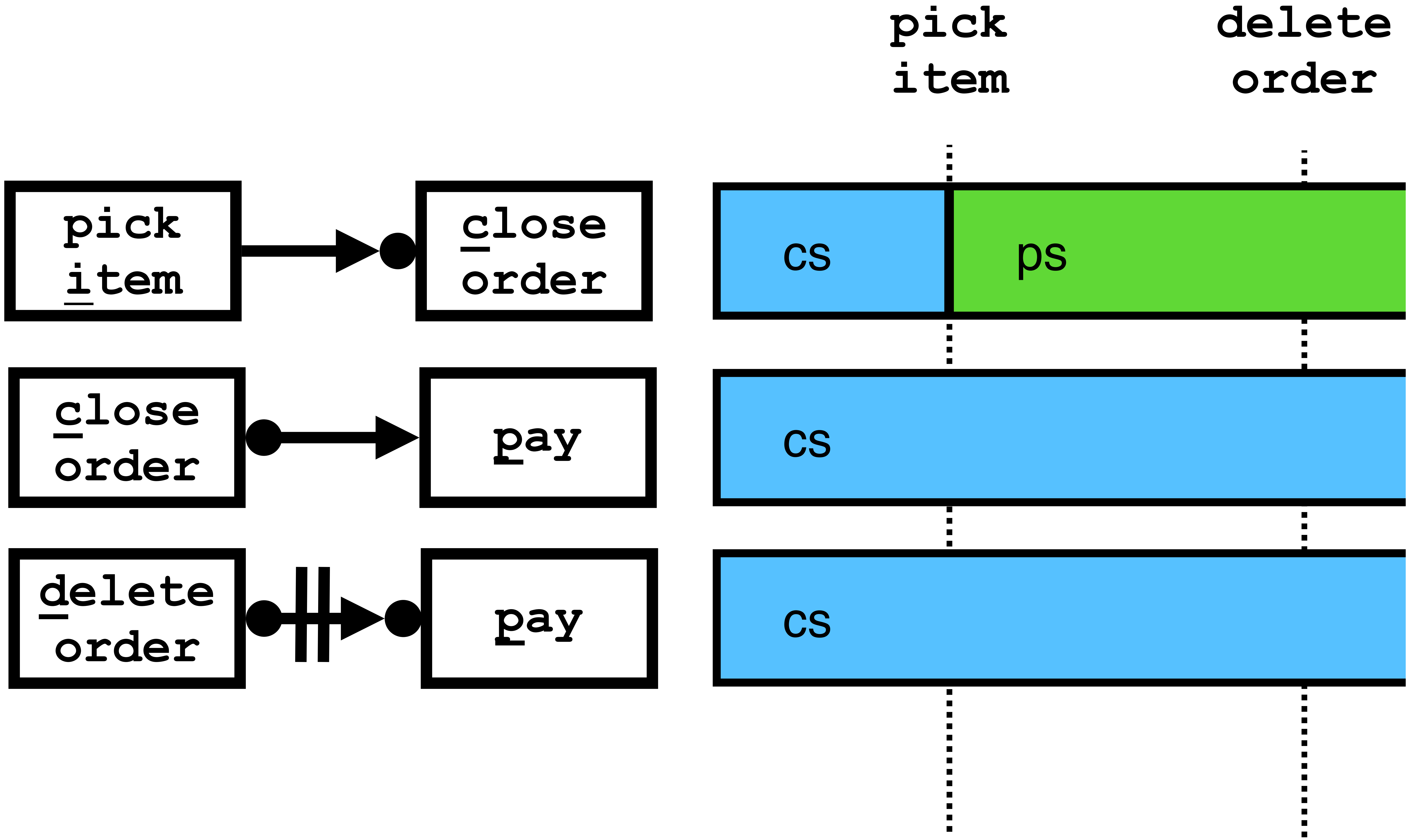
Monitor in action



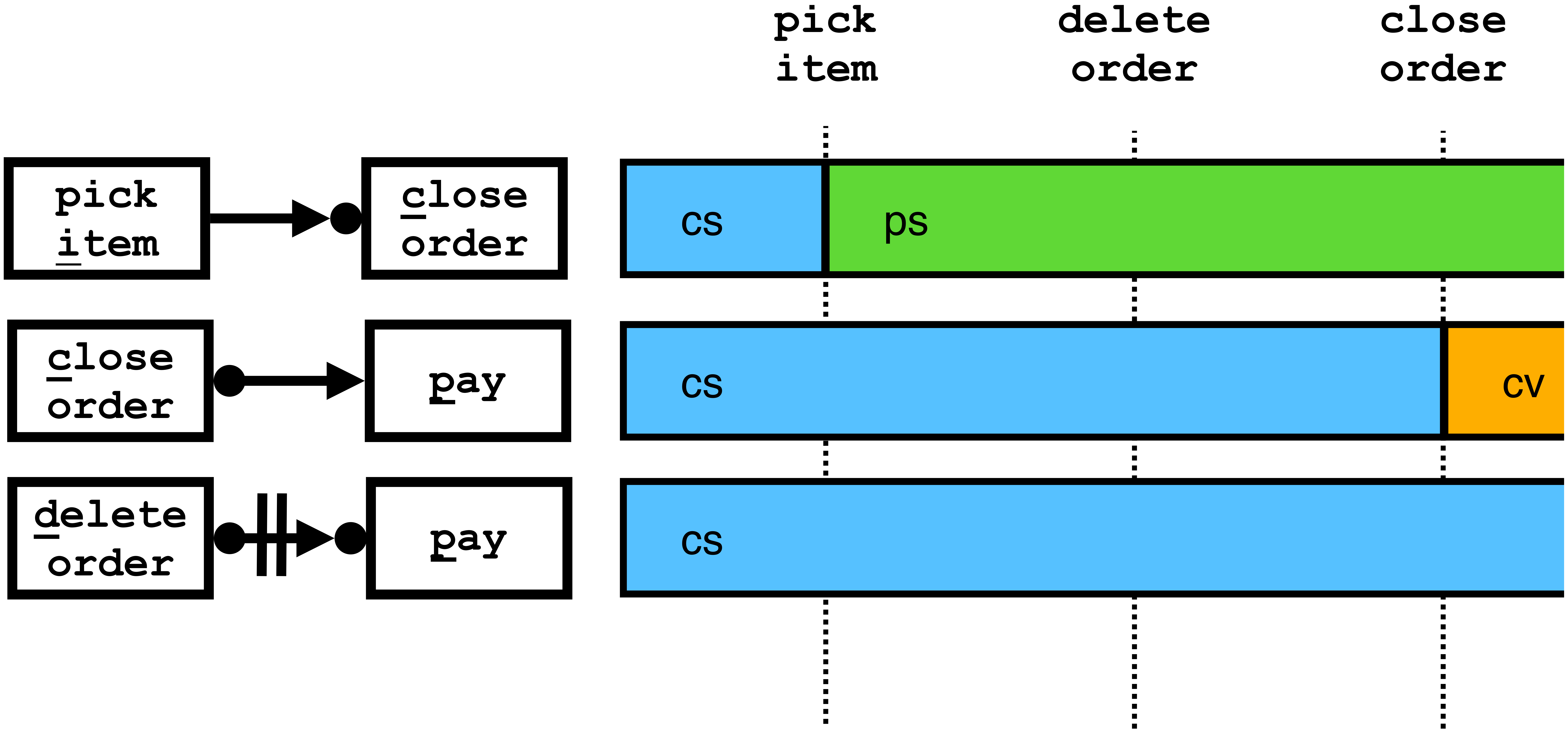
Monitor in action



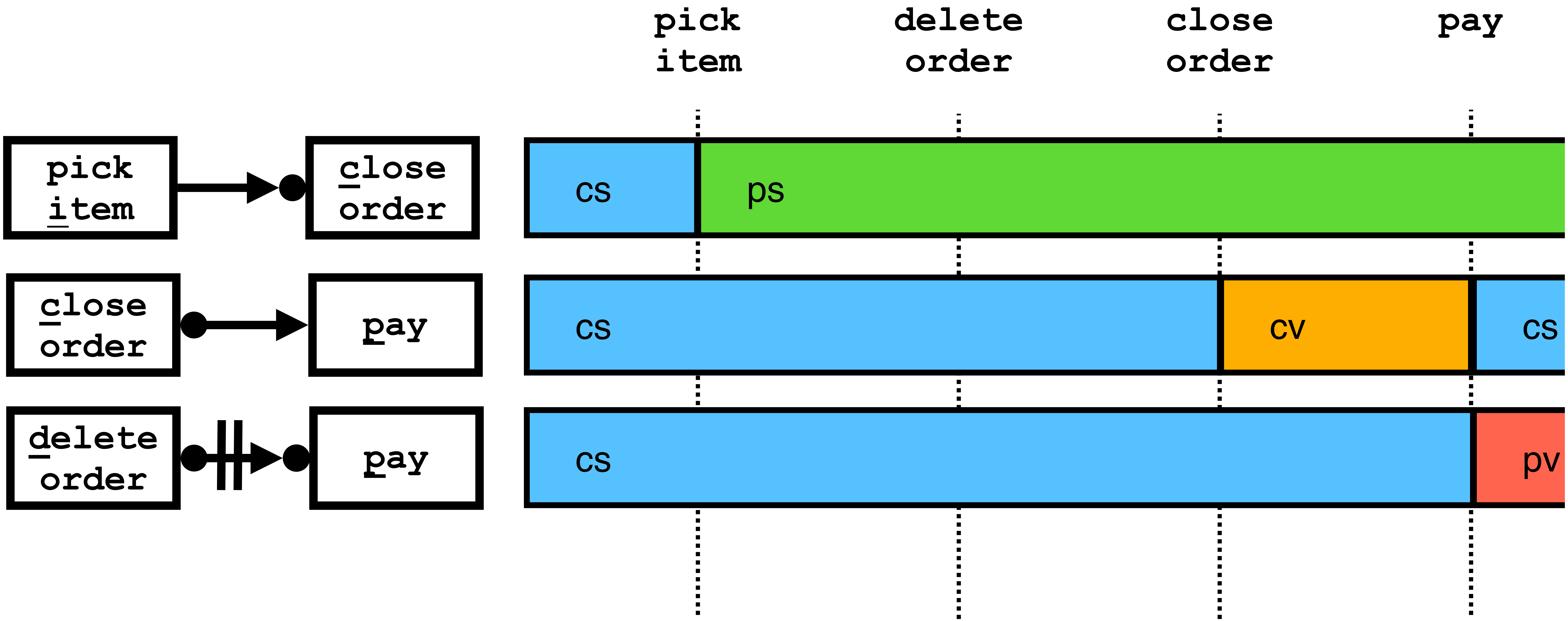
Monitor in action



Monitor in action



Monitor in action



Monitor in action

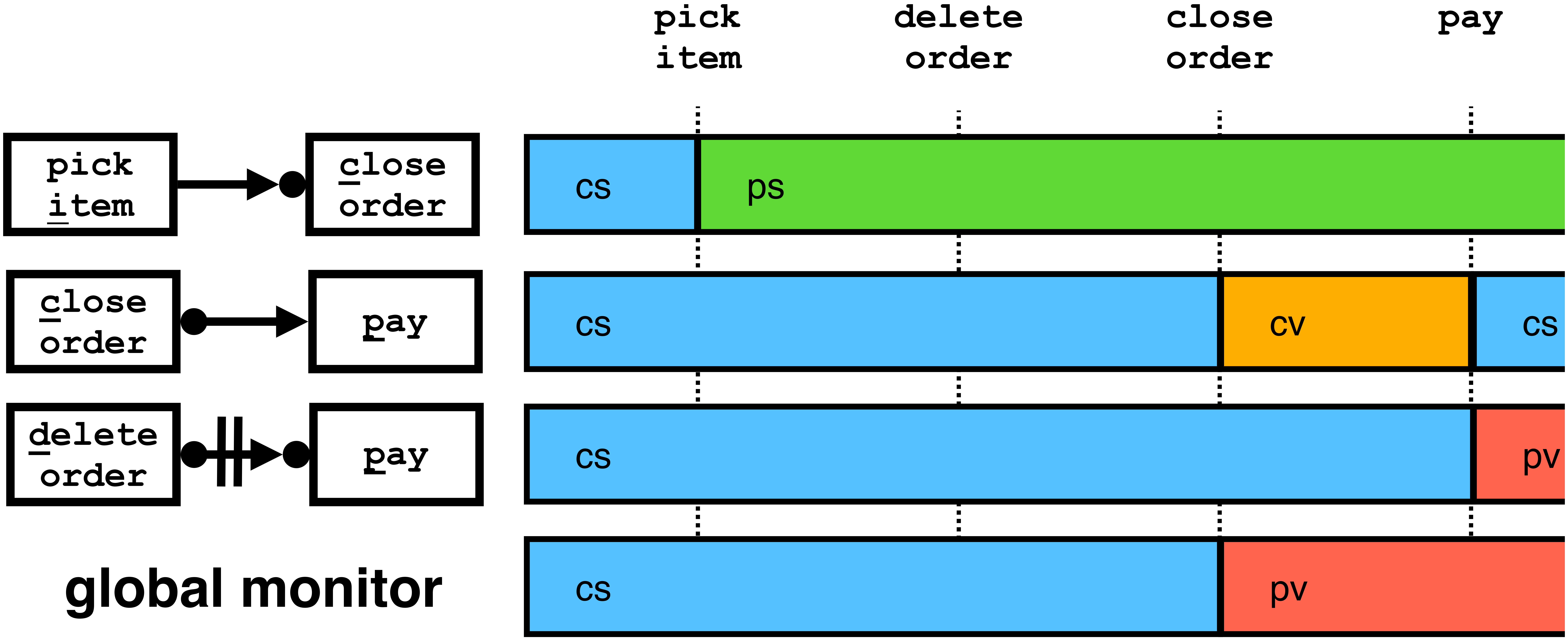
pick item
delete order
close order
pay
pick item
close order
cs
ps
close order
pay
cs
cv
cs
delete order
pay
cs
pv

Quiz: is this the earliest instant for detecting a violation?

Monitor in action

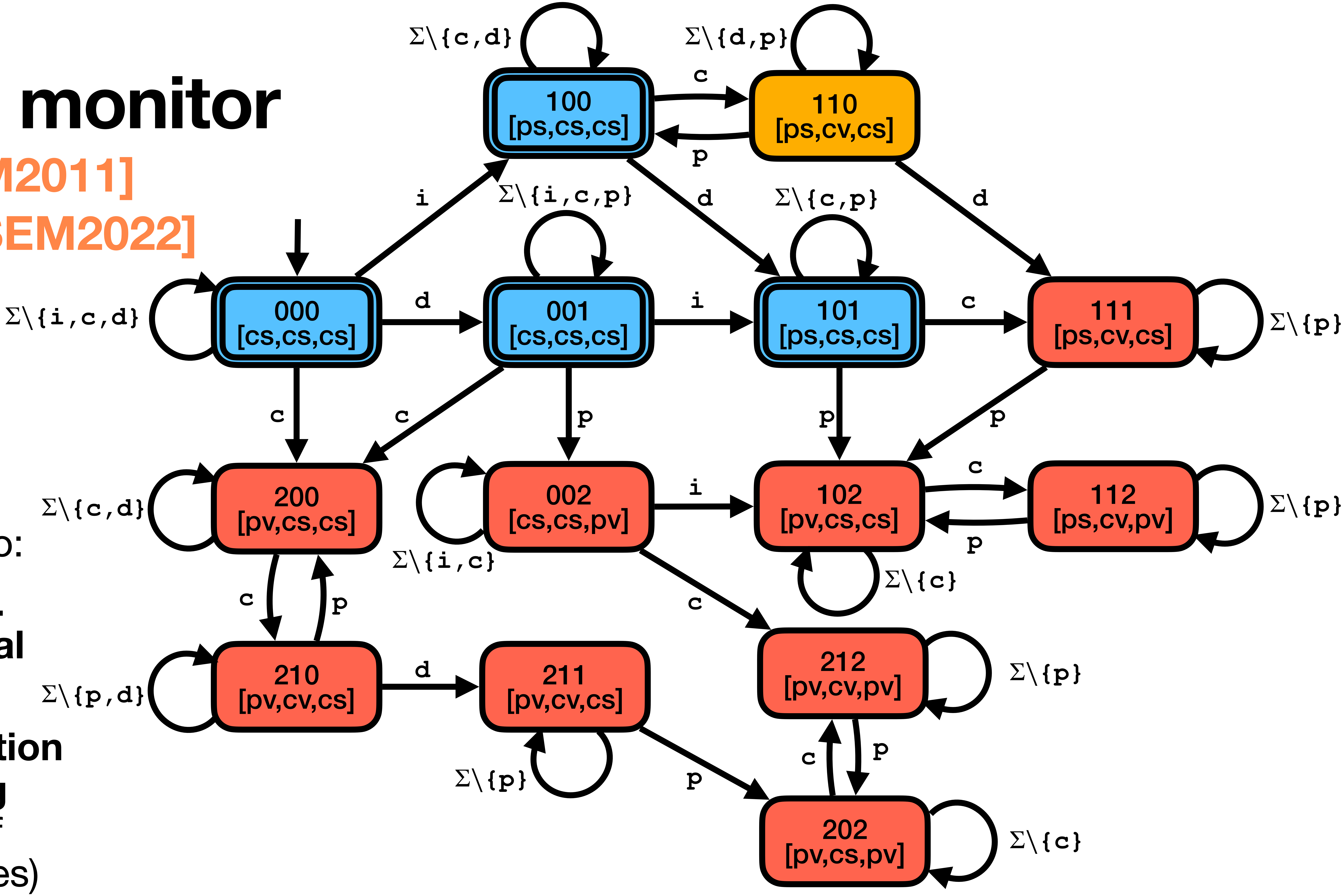
pick
item
delete
order
close
order
pay
pick
item
close
order
close
order
pay
delete
order
pay
cs
cv
cs
cs
pv
X
To satisfy: pay
To keep satisfied: don’t pay

Monitor in action



Global monitor

[____,BPM2011]
[____,TOSEM2022]

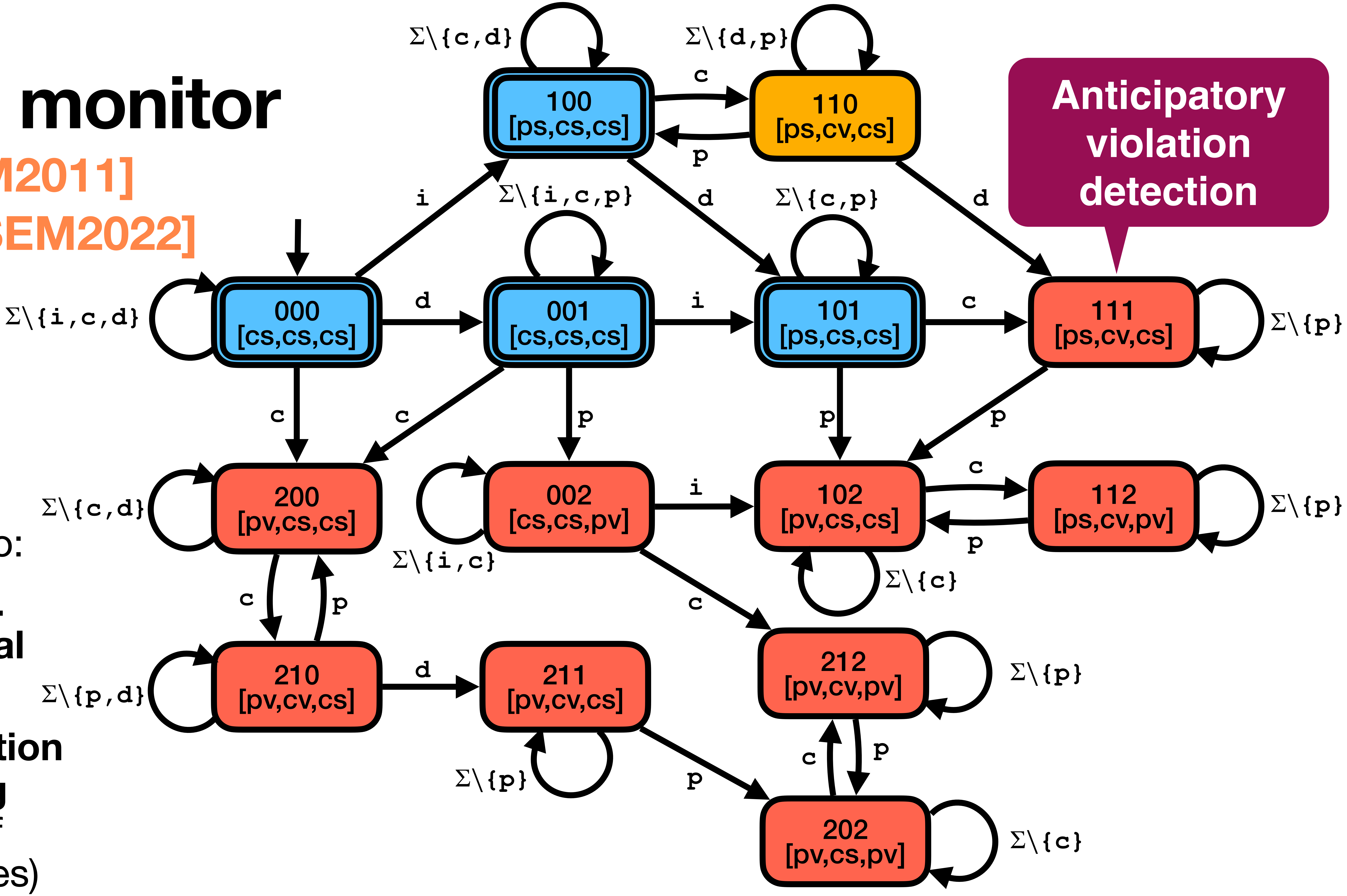


Cross-product with two proviso:

- recall **RV-LTL labels of local constraints**
- **no minimisation nor trimming** (distinction of violation states)

Global monitor

[____,BPM2011]
[____,TOSEM2022]



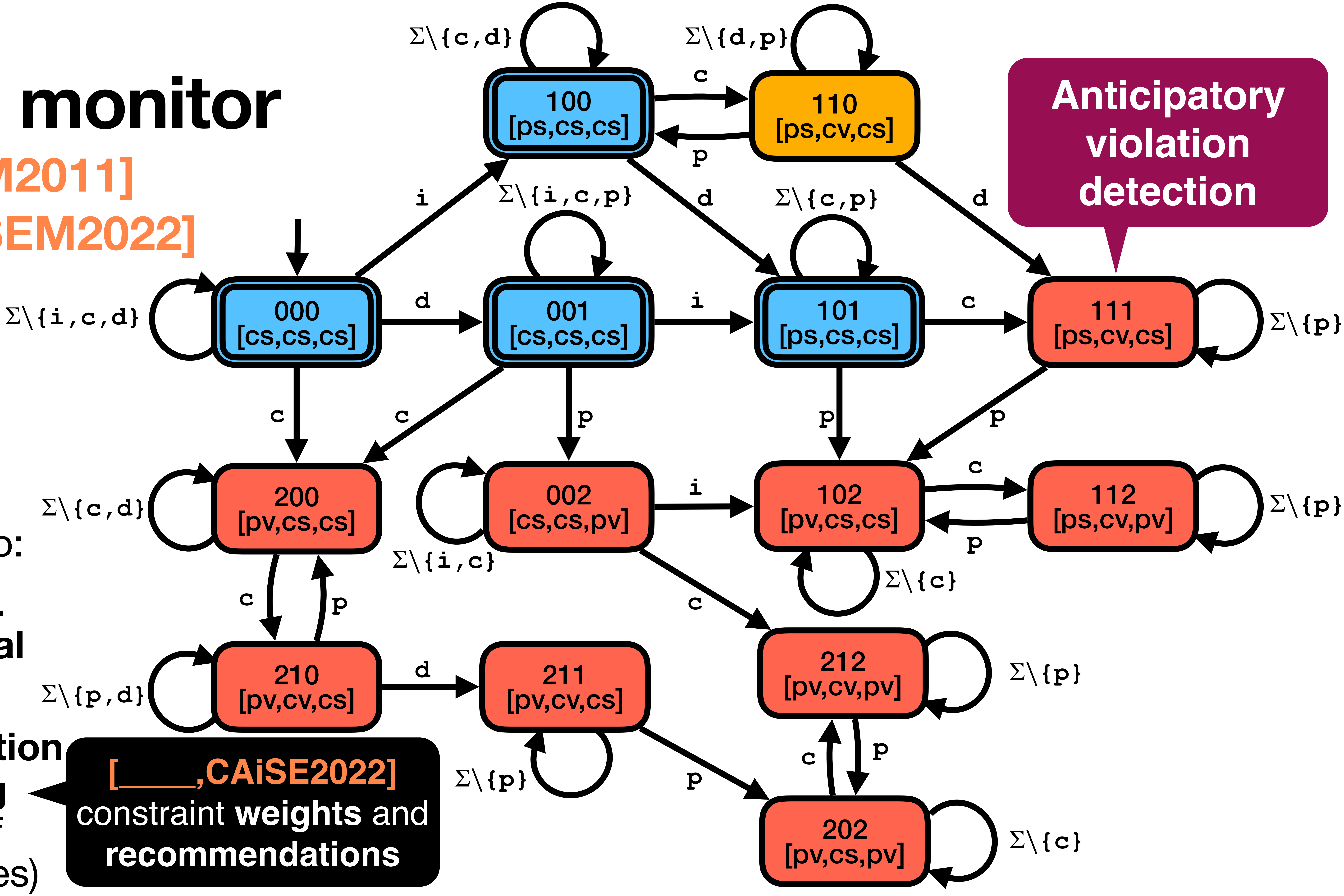
Anticipatory violation detection

Cross-product with two proviso:

- recall **RV-LTL labels of local constraints**
- **no minimisation nor trimming** (distinction of violation states)

Global monitor

[____,BPM2011]
[____,TOSEM2022]



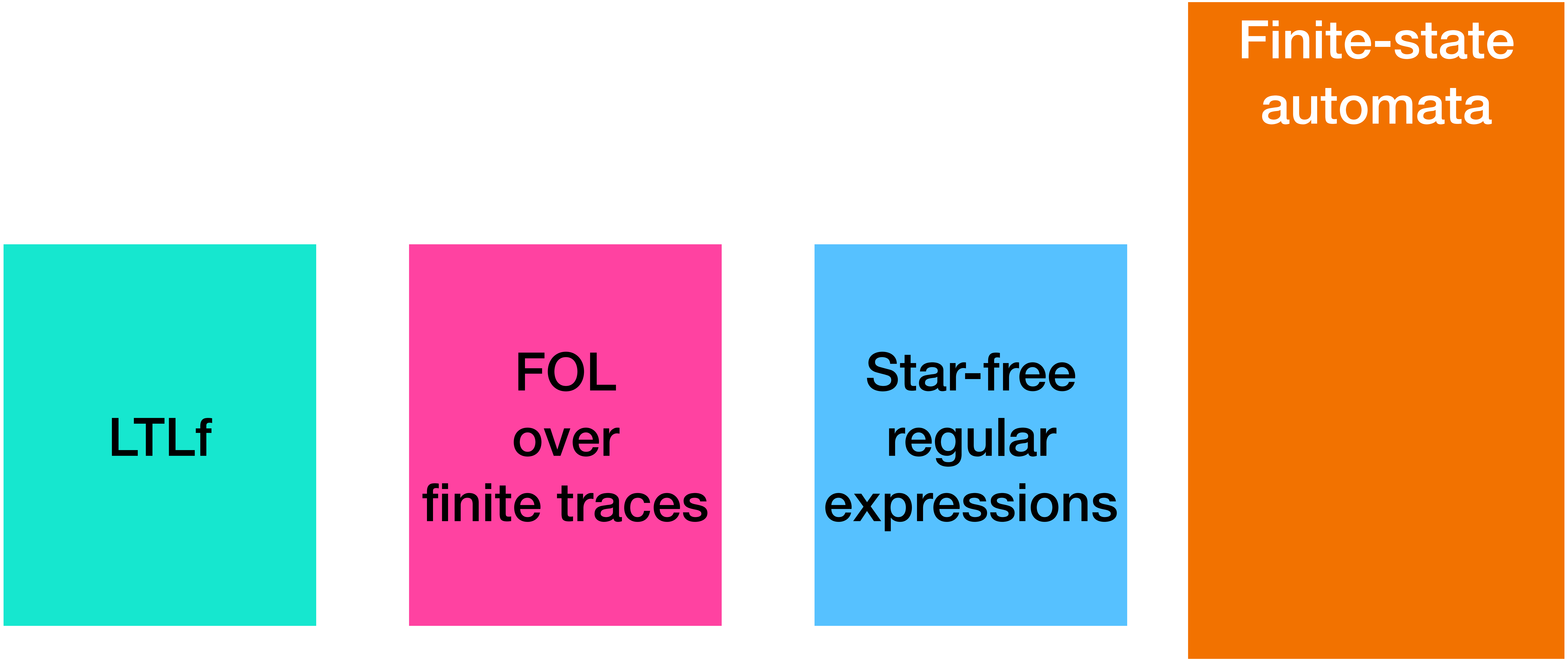
Anticipatory violation detection

Cross-product with two proviso:

- recall **RV-LTL labels of local constraints**
- **no minimisation nor trimming** (distinction of violation states)

[____,CAiSE2022]
constraint **weights** and **recommendations**

Can we do more?



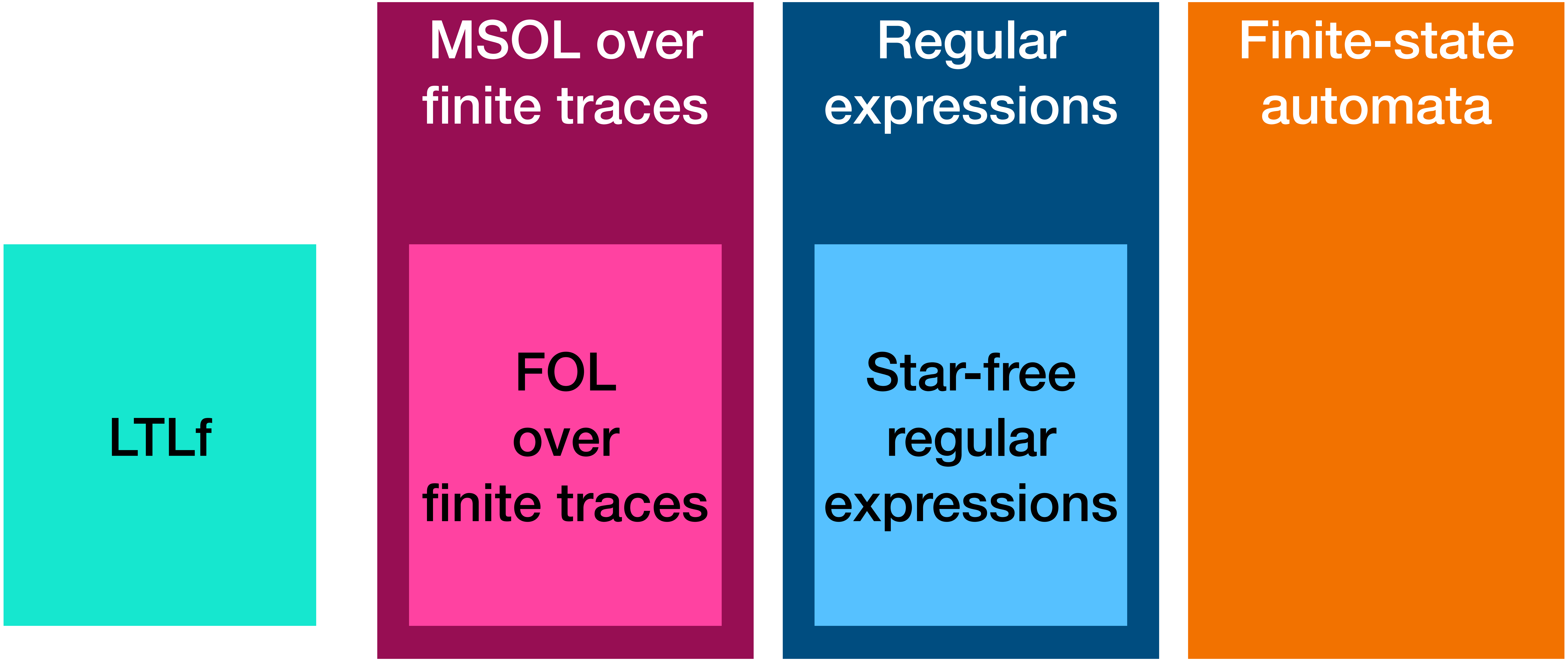
LTLf

FOL over finite traces

Star-free regular expressions

Finite-state automata

Can we do more?



LTLf

MSOL over finite traces

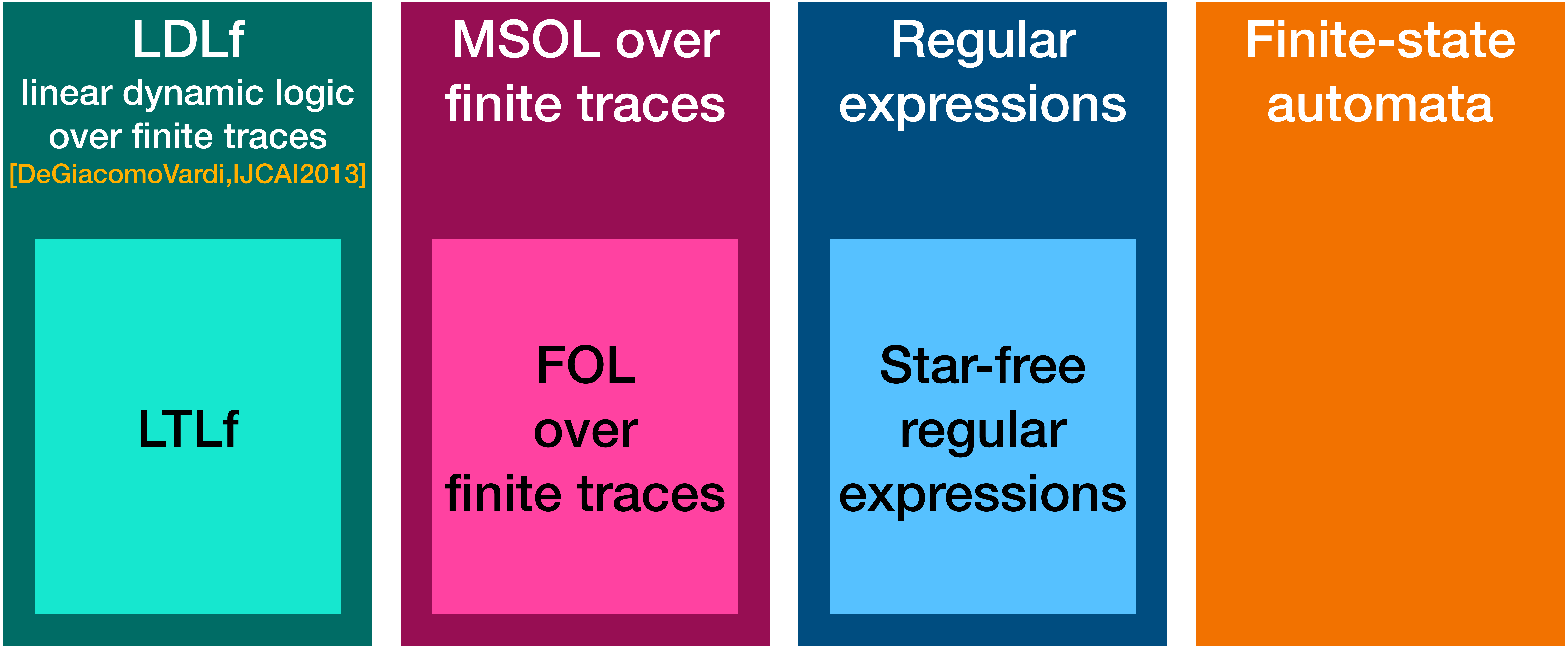
FOL over finite traces

Regular expressions

Star-free regular expressions

Finite-state automata

Can we do more?



LDLf

linear dynamic logic over finite traces

[DeGiacomoVardi,IJCAI2013]

LTLf

MSOL over finite traces

FOL over finite traces

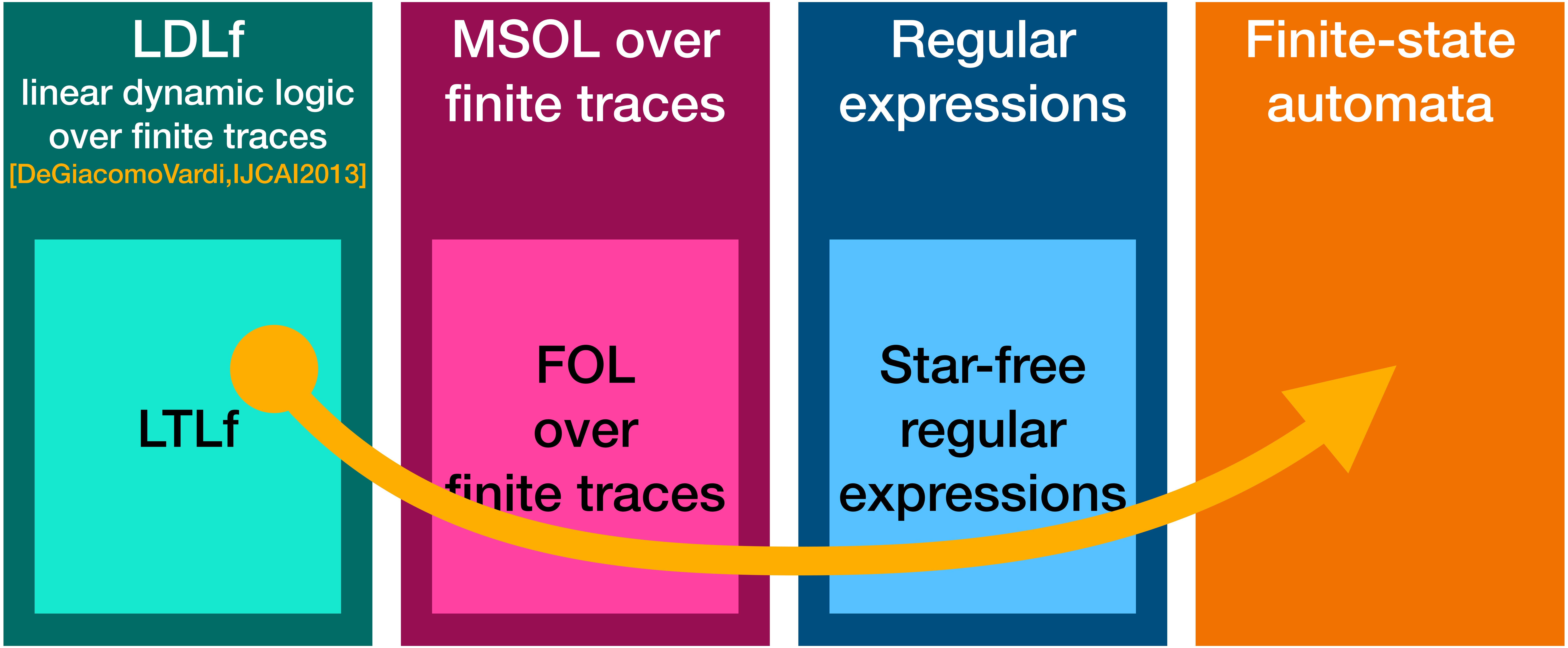
Regular expressions

Star-free regular expressions

Finite-state automata

Can we do more?

[____,BPM2014] [____,TOSEM2022]



LDLf

linear dynamic logic over finite traces

[DeGiacomoVardi,IJCAI2013]

LTLf

MSOL over finite traces

FOL over finite traces

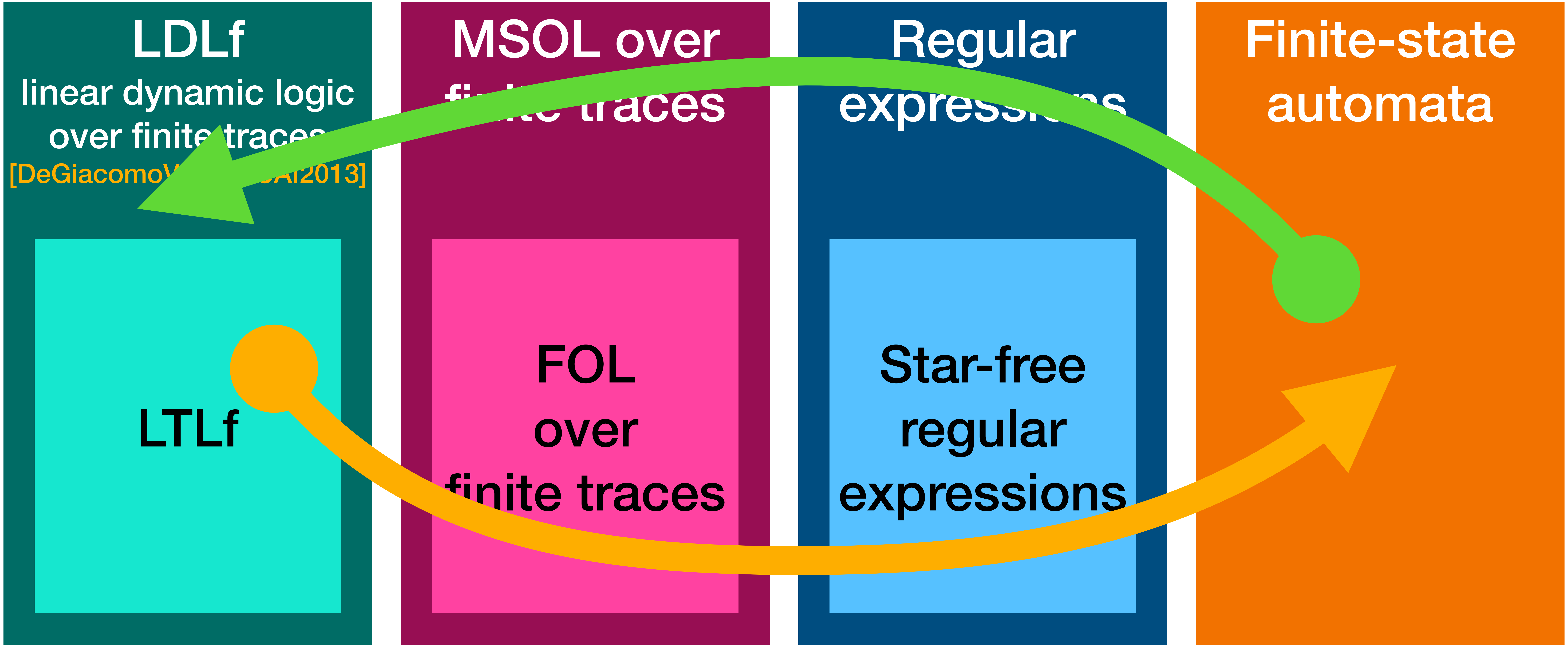
Regular expressions

Star-free regular expressions

Finite-state automata

Can we do more?

[____,BPM2014] [____,TOSEM2022]



LDLf

linear dynamic logic over finite traces

[DeGiacomoVardi,IJCAI2013]

LTLf

MSOL over finite traces

FOL over finite traces

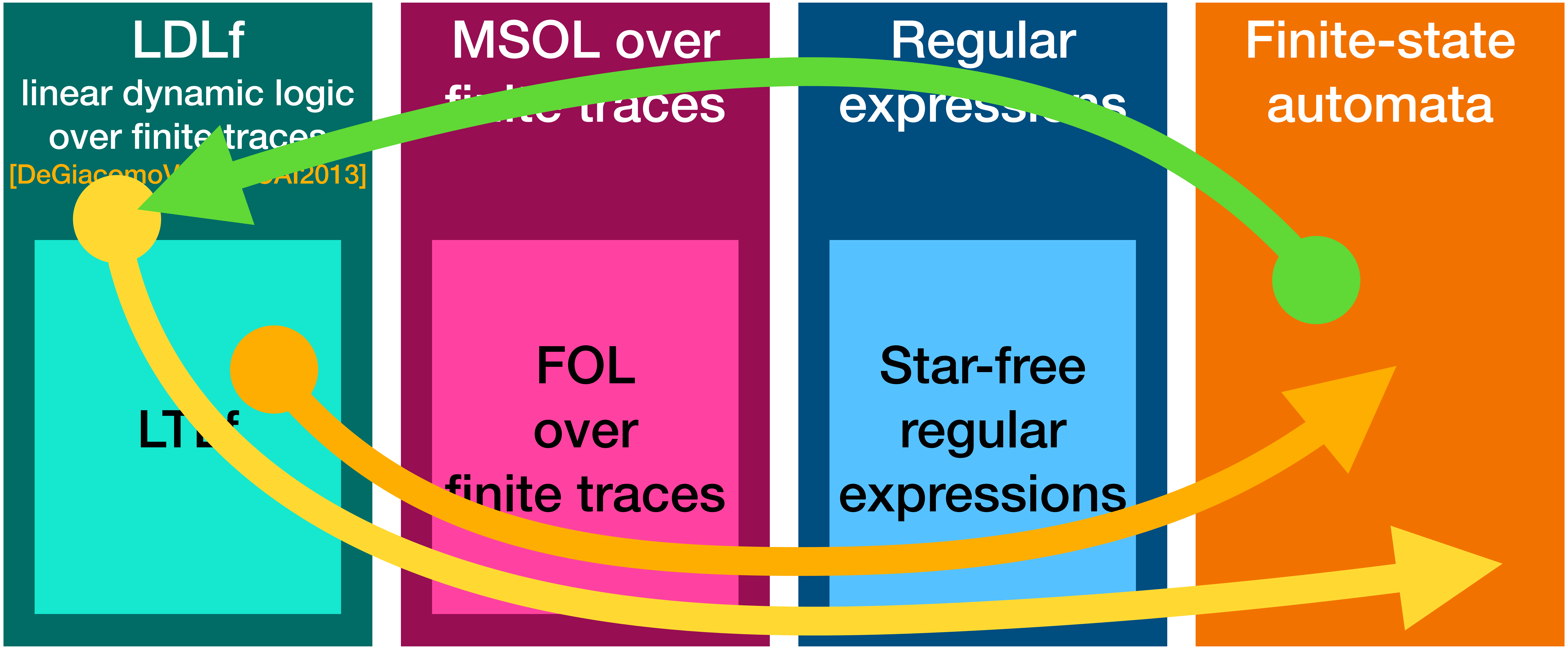
Regular expressions

Star-free regular expressions

Finite-state automata

Can we do more?

[____,BPM2014] [____,TOSEM2022]



LDLf

linear dynamic logic over finite traces

[DeGiacomoVardi,IJCAI2013]

LTLf

MSOL over finite traces

FOL over finite traces

Regular expressions

Star-free regular expressions

Finite-state automata

From constraints to metaconstraints

[____,BPM2014] [____,TOSEM2022]

LDLf expresses RV-LTLf monitoring states of LDLf constraints

- Support for **metaconstraints predicating over the monitoring status of other constraints**

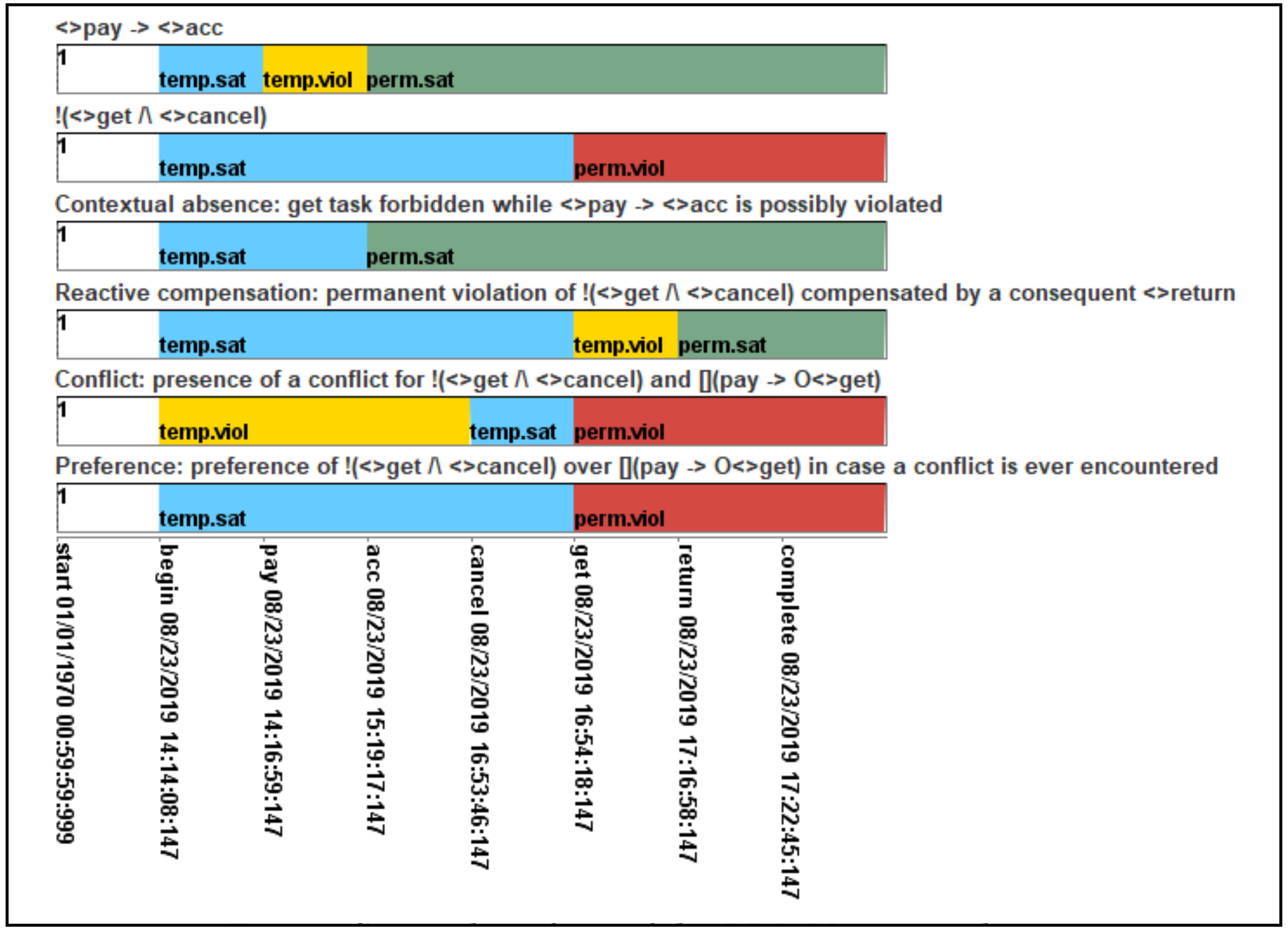
Example: a form of **“contrary-to-duty” process constraint**

- If **constraint C1** gets **permanently violated**, eventually **satisfy** a **compensation constraint C2**

Interesting open problem: relationship with **normative frameworks** and **defeasible reasoning**

Tooling

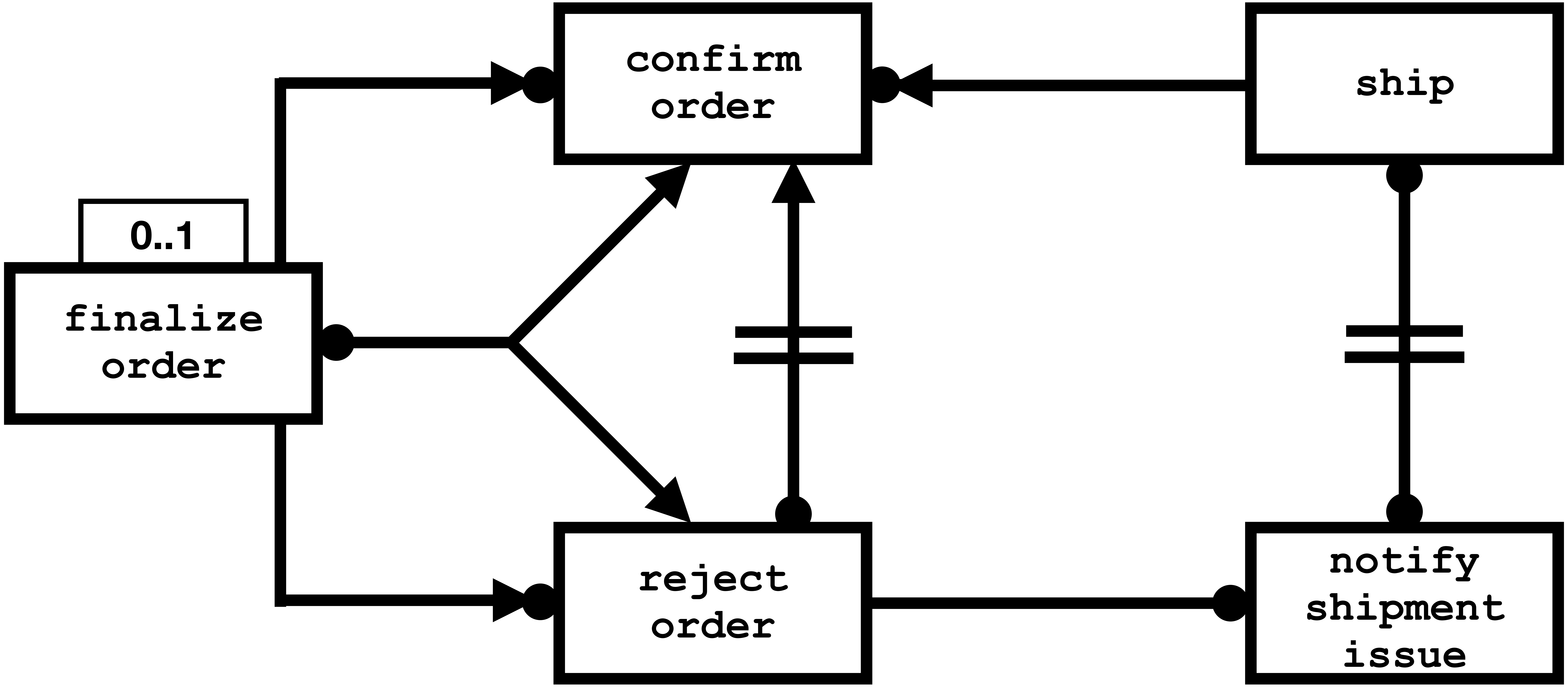
Fully implemented as part of the **RuM toolkit** (rulemining.org)



From global monitor to enactment

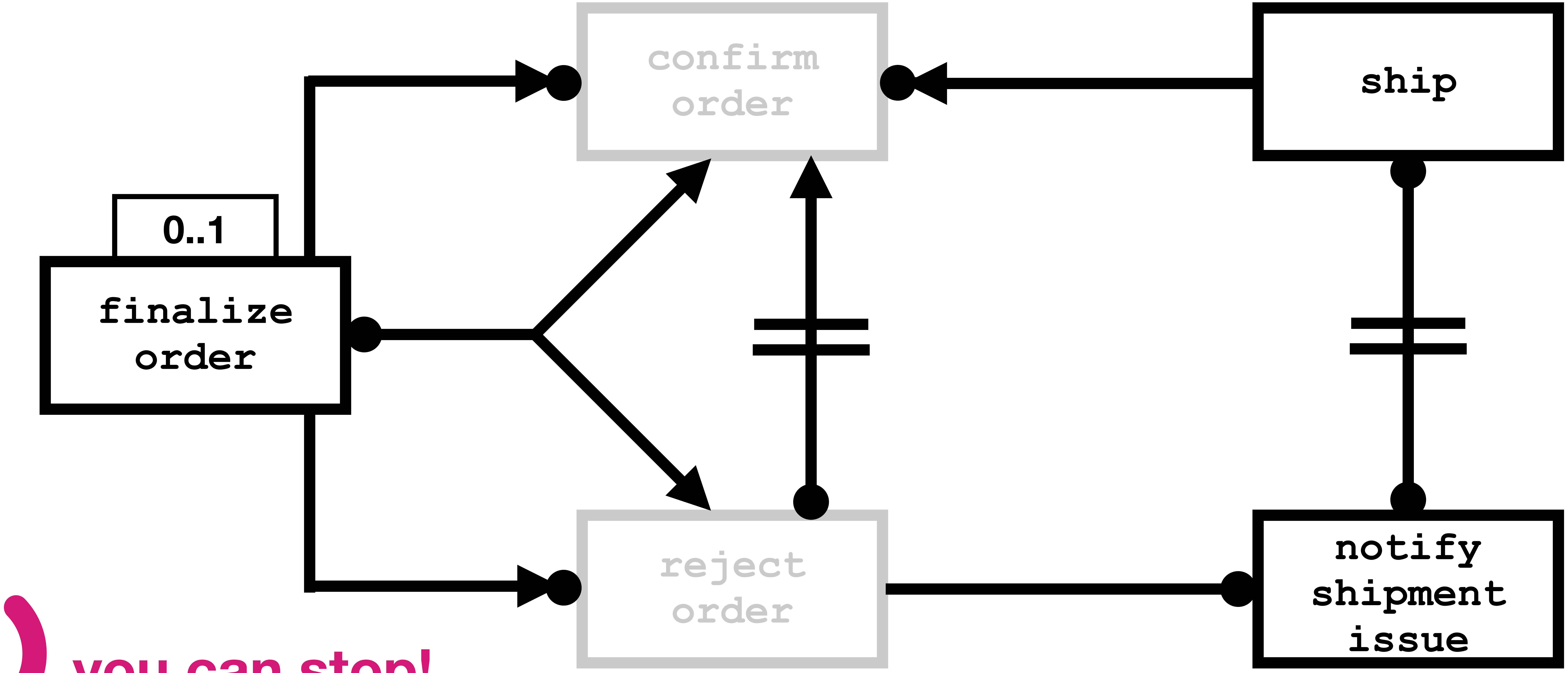
1. Compute the global, colored DFA **A**
2. **s** = initial state of **A**
3. Loop
    A. **Block** all tasks that would lead to a **permanent violation** if executed in **s**
    B. **Highlight** constraints that are **permanently satisfied** in **s**
    C. **Highlight** constraints that are **currently violated**
    If no currently violated constraint: allow for **completing** the process
    D. **Use picks** an enabled task **a** and **executes** it
    E. Fetch **s'** s.t. **<s,a,s'>** belongs to **A**
    F. **s = s'**

Example


confirm order
ship
0..1
finalize order
reject order
notify shipment issue

Example

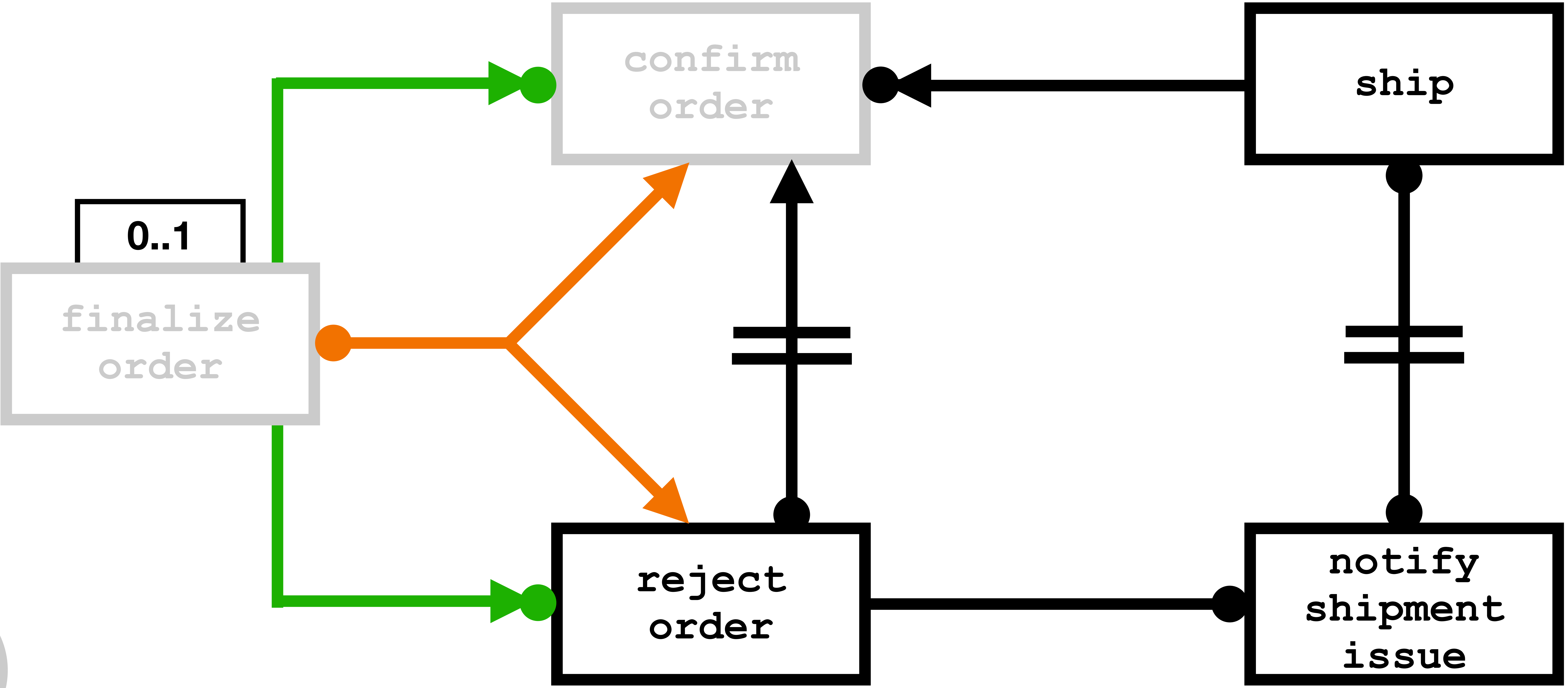
Initial state



you can stop!

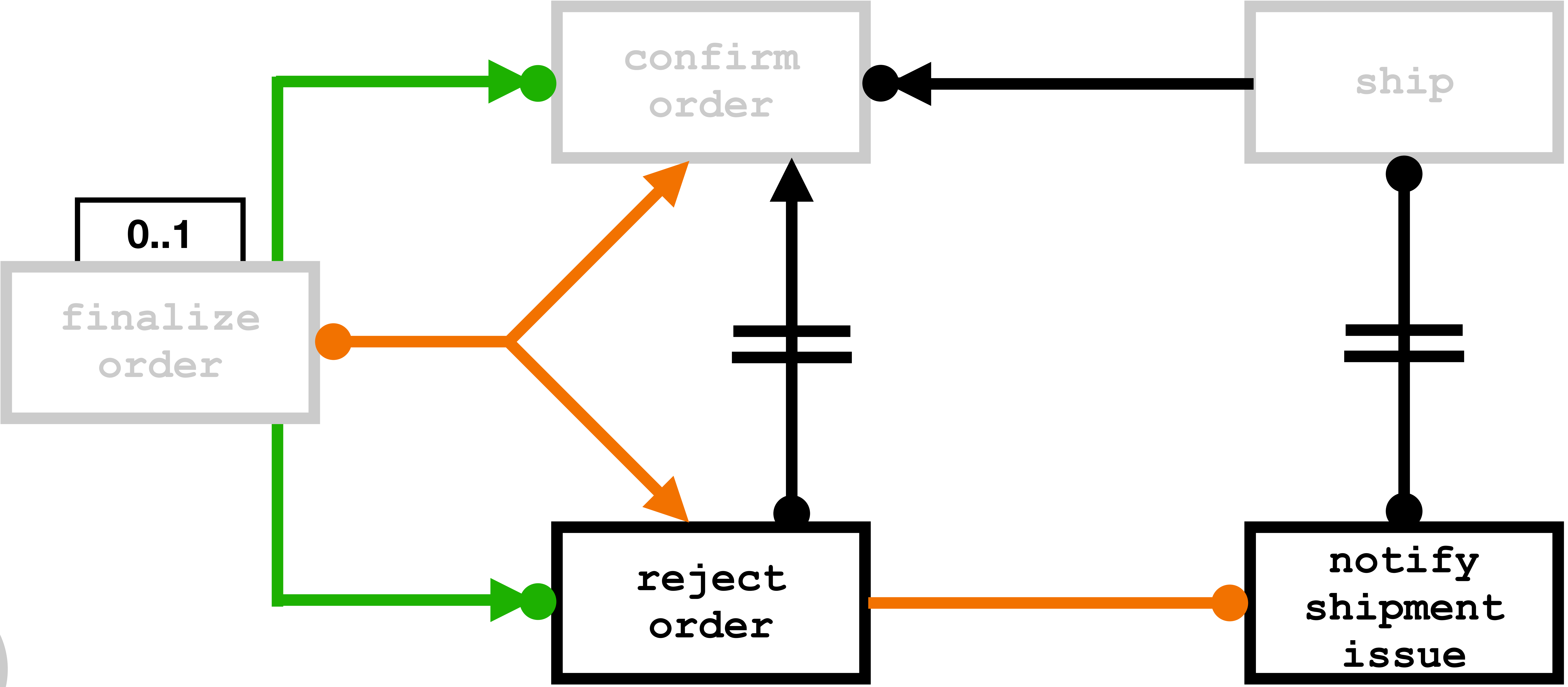
Example

“finalize order”



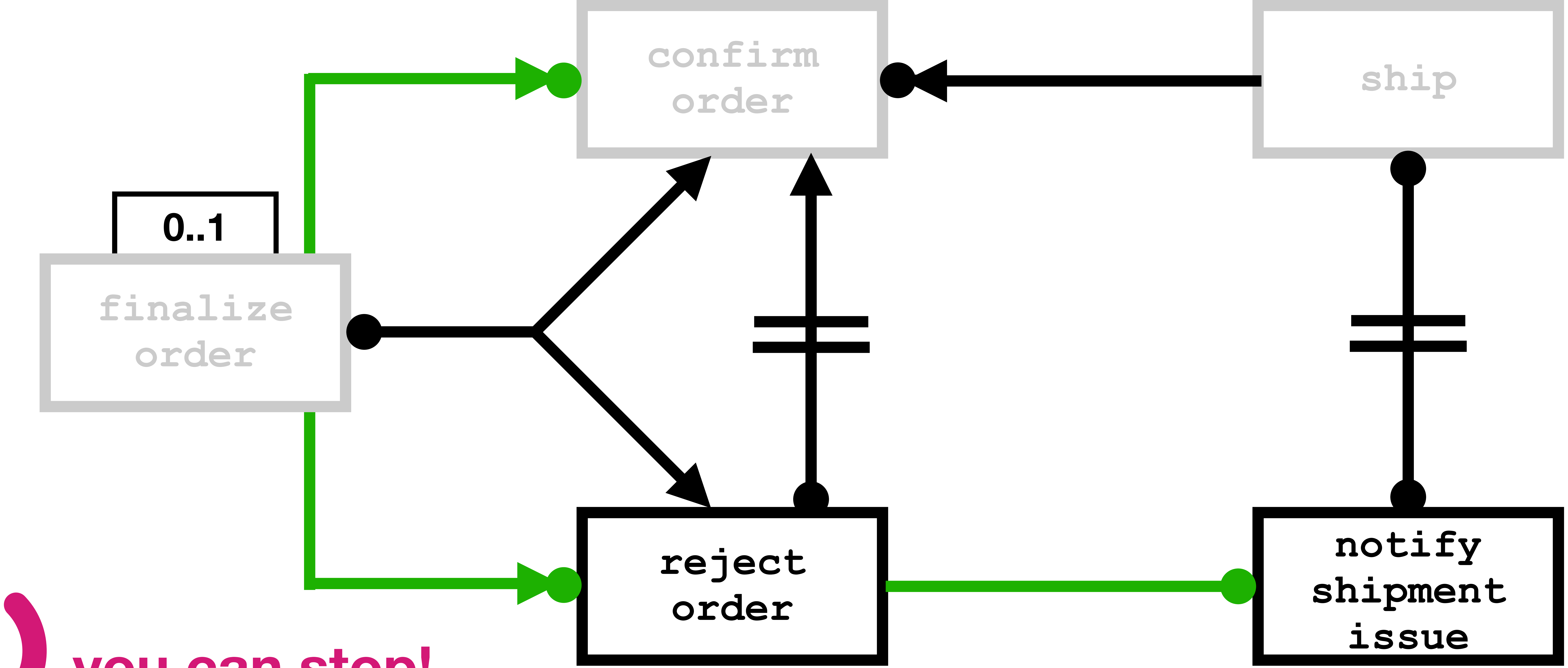
Example

“notify shipment issue”



Example

“reject order”



you can stop!

Declarative process discovery

Declarative process discovery

Simply stated…

Process discovery aiming at extracting a **declarative specification** from a log

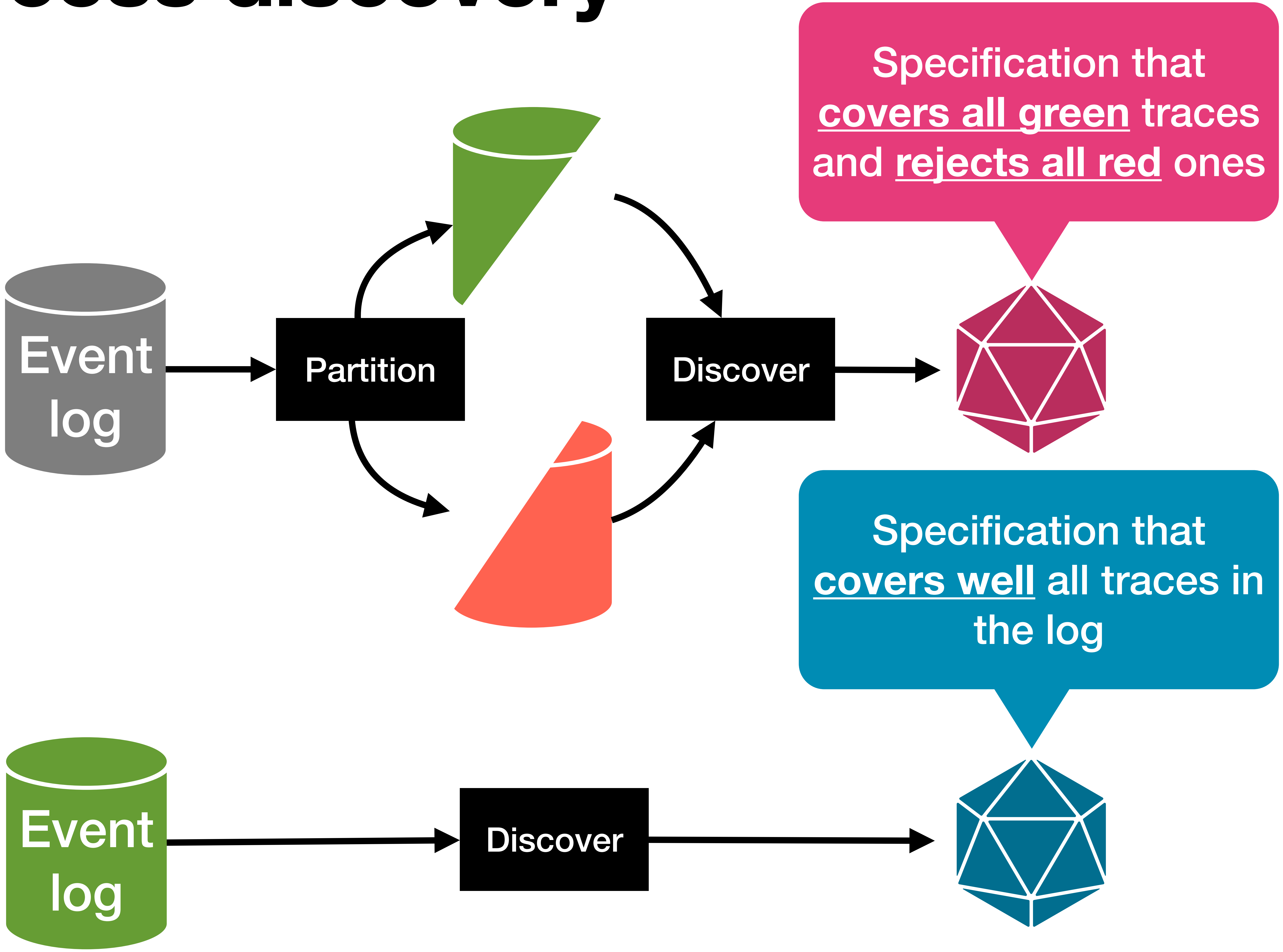
In our case: Declare

Declarative process discovery

Two settings

Discriminative mining

Specification mining



Specification that **covers all green** traces and **rejects all red** ones

Specification that **covers well** all traces in the log

All possible constraints grounded on the activities in the log

Which to keep?

All possible constraints grounded on the activities in the log

General idea

1. Define suitable **metrics** to capture the meaning of **covering well** -> **interesting satisfaction**
   - Starting point: **support** and **confidence** from data mining
   - Issues: **not enough**, **not easy to import** (see next slides)
2. Approach **discovery by combining**:
   - **Metrics** for interestingness
   - Temporal **reasoning** for logical correctness and for computing the inputs of metrics

Naive support is not enough

$$\text{Constraint support (naive)} = \frac{\text{\# traces that satisfy the constraint}}{\text{total \# traces in the log}}$$

Issue: consider **response(a,b)**

- <a,c,b,a>
- <c,d,e,c,d,e,f>
- <b,c,d,e>
- <a,b>
- <a,a,b,a,b,a,b,a,a,a,b>

Naive support is not enough

$$\text{Constraint support (naive)} = \frac{\text{\# traces that satisfy the constraint}}{\text{total \# traces in the log}}$$

Issue: consider **response(a,b)**

- <a,c,b,a>
- <c,d,e,c,d,e,f>
- <b,c,d,e>
- <a,b>
- <a,a,b,a,b,a,b,a,a,a,b>

Support: 4/5 (**not informative**)

Need to:

1. Account for **vacuous satisfaction**
2. Distinguish satisfying traces based on **interestingness**
3. Define **event-based measures**

In search of a normal form

$$\Box(\psi \to \varphi)$$

In search of a normal form

$$\Box(\psi \rightarrow \varphi)$$

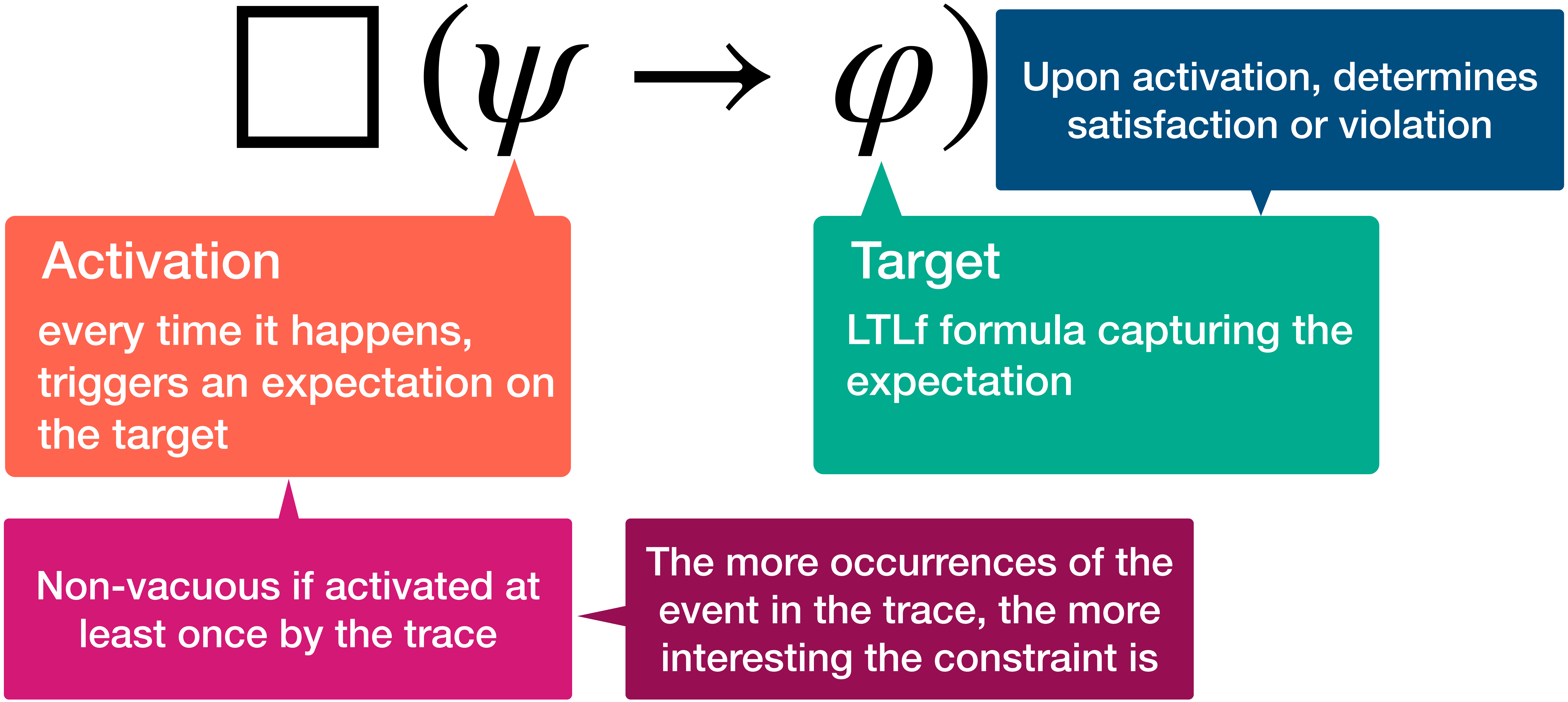
Activation

every time it happens, triggers an expectation on the target

Target

LTLf formula capturing the expectation

In search of a normal form



$$\Box(\psi \rightarrow \varphi)$$

Upon activation, determines satisfaction or violation

Activation

every time it happens, triggers an expectation on the target

Target

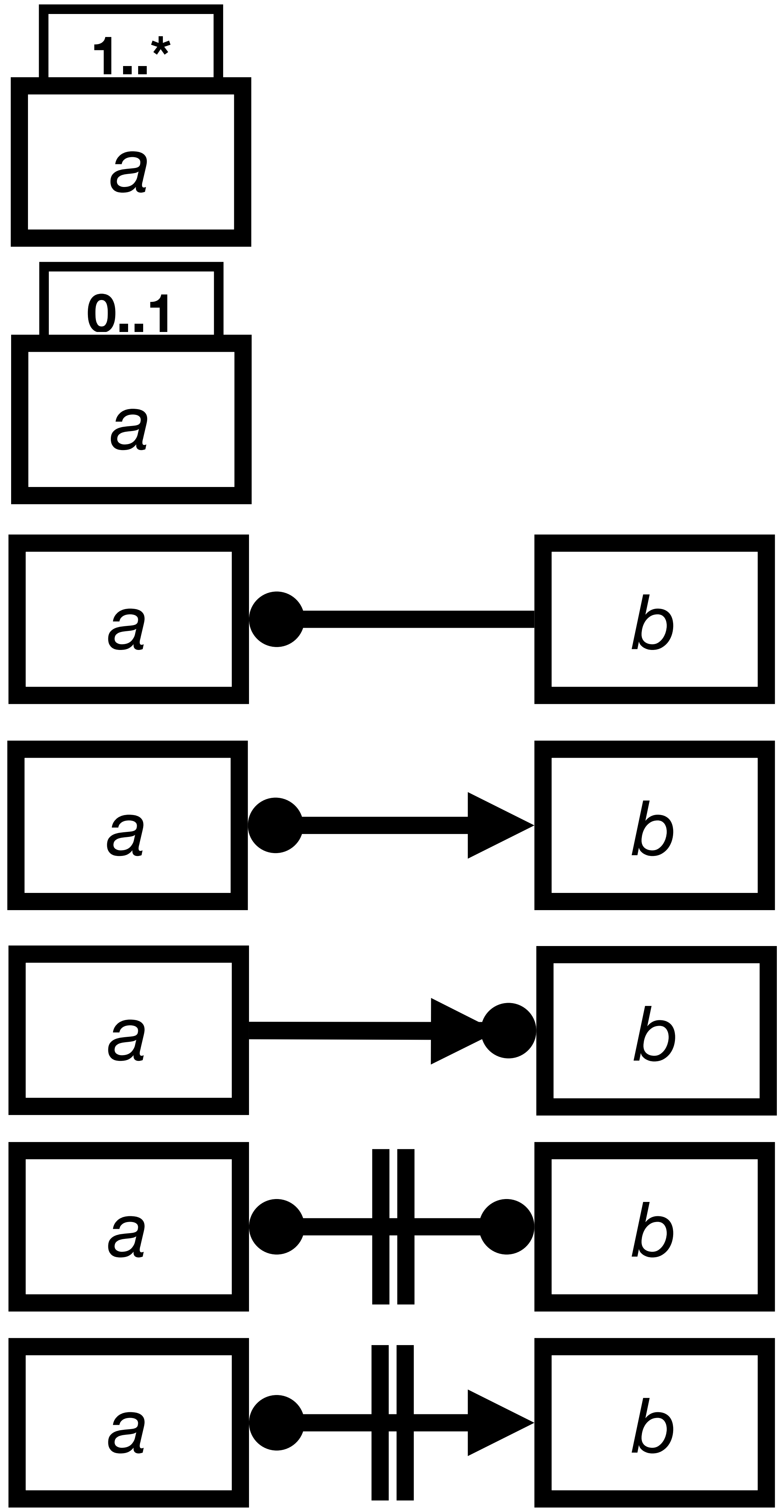
LTLf formula capturing the expectation

Non-vacuous if activated at least once by the trace

The more occurrences of the event in the trace, the more interesting the constraint is

Redefining templates…

(exploiting past-tense operators)



activation target

$\Diamond a$ → $\Box(\quad \rightarrow \quad)$

$\neg\Diamond(a \wedge \mathsf{O}\Diamond a)$ → $\Box(\quad \rightarrow \quad)$

$\Diamond a \rightarrow \Diamond b$ → $\Box(\quad \rightarrow \quad)$

$\Box(a \rightarrow \mathsf{O}\Diamond b)$ → $\Box(\quad \rightarrow \quad)$

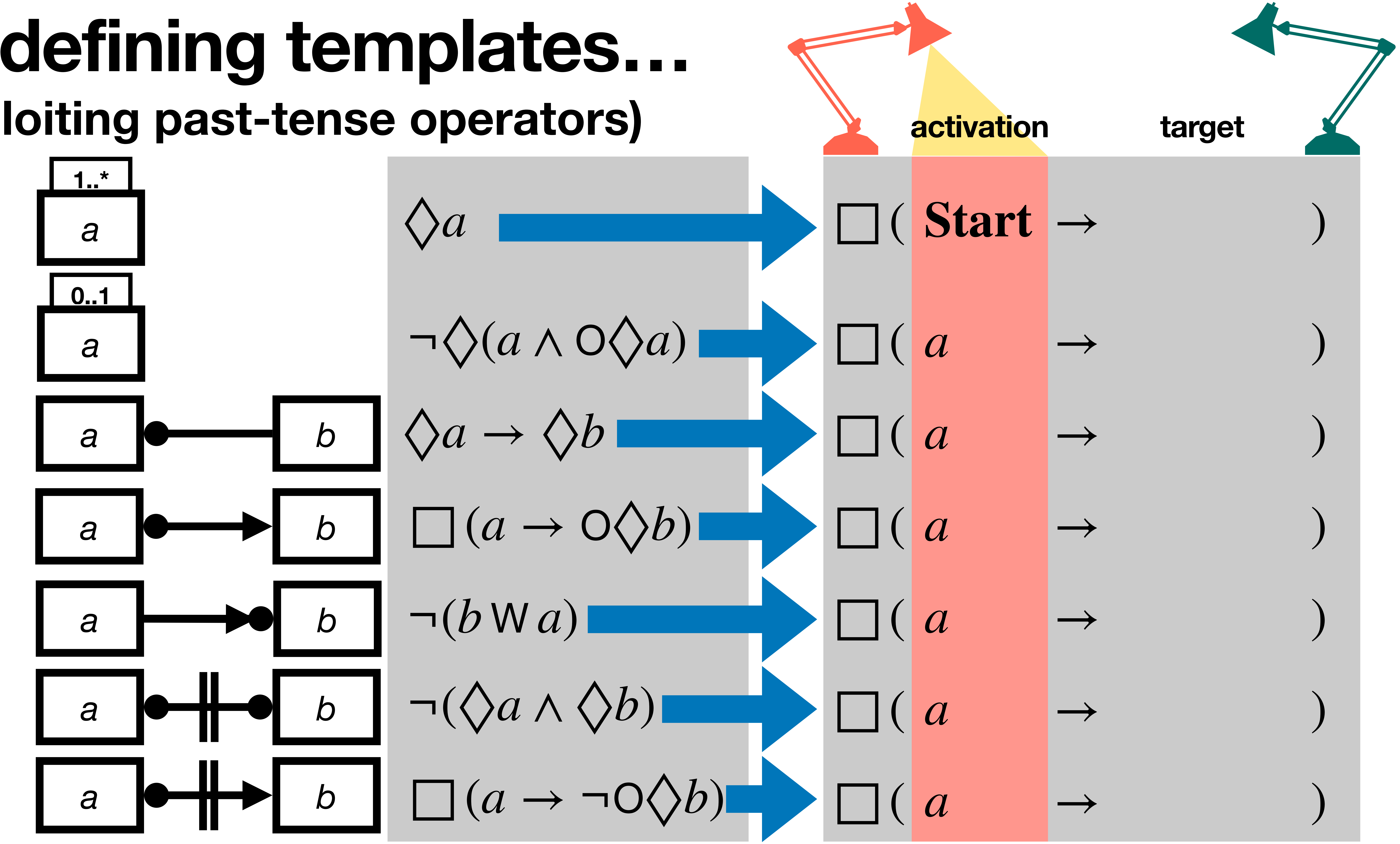
$\neg(b \, \mathsf{W} \, a)$ → $\Box(\quad \rightarrow \quad)$

$\neg(\Diamond a \wedge \Diamond b)$ → $\Box(\quad \rightarrow \quad)$

$\Box(a \rightarrow \neg\mathsf{O}\Diamond b)$ → $\Box(\quad \rightarrow \quad)$

Redefining templates…

(exploiting past-tense operators)



$\Diamond a$ → $\Box(\text{Start} \rightarrow \quad)$

$\neg\Diamond(a \wedge \bigcirc\Diamond a)$ → $\Box(a \rightarrow \quad)$

$\Diamond a \rightarrow \Diamond b$ → $\Box(a \rightarrow \quad)$

$\Box(a \rightarrow \bigcirc\Diamond b)$ → $\Box(a \rightarrow \quad)$

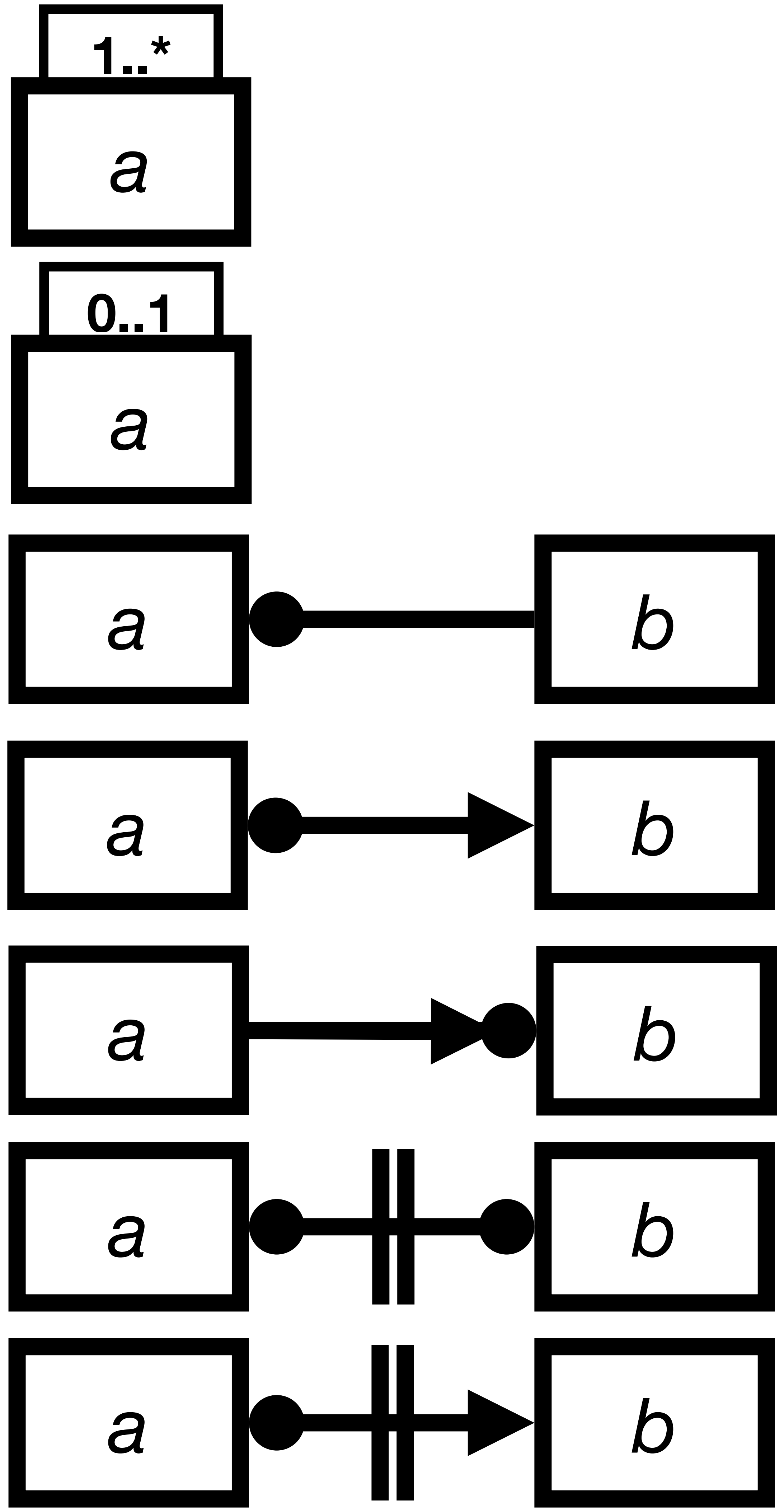
$\neg(b \,\mathsf{W}\, a)$ → $\Box(a \rightarrow \quad)$

$\neg(\Diamond a \wedge \Diamond b)$ → $\Box(a \rightarrow \quad)$

$\Box(a \rightarrow \neg\bigcirc\Diamond b)$ → $\Box(a \rightarrow \quad)$

Redefining templates…

(exploiting past-tense operators)



		activation		target	
$\Diamond a$	$\Box($	**Start**	$\rightarrow$	$\Diamond a$	$)$
$\neg\Diamond(a \land \bigcirc\Diamond a)$	$\Box($	a	$\rightarrow$	$\neg\bigcirc\Diamond a$	$)$
$\Diamond a \rightarrow \Diamond b$	$\Box($	a	$\rightarrow$	$\Diamond b \lor \diamondminus b$	$)$
$\Box(a \rightarrow \bigcirc\Diamond b)$	$\Box($	a	$\rightarrow$	$\Diamond b$	$)$
$\neg(b \,\mathsf{W}\, a)$	$\Box($	b	$\rightarrow$	$\diamondminus a$	$)$
$\neg(\Diamond a \land \Diamond b)$	$\Box($	a	$\rightarrow$	$\Box\neg b$	$)$
$\Box(a \rightarrow \neg\bigcirc\Diamond b)$	$\Box($	a	$\rightarrow$	$\neg\bigcirc\Diamond b)$	$)$

Trace-based support, refined

$$\text{Constraint support (trace-based)} = \frac{\text{\# traces that satisfy the constraint and activate it}}{\text{total \# traces in the log}}$$

Response(a,b)

- <a,c,b,a>
- <c,d,e,c,d,e,f>
- <b,c,d,e>
- <a,c,d,b>
- <a,a,c,b,a,b,a,d,b,a,a,c,a,b>

Trace-based support, refined

$$\text{Constraint support (trace-based)} = \frac{\text{\# traces that satisfy the constraint and activate it}}{\text{total \# traces in the log}}$$

Response(a,b)

- <a,c,b,a>
- <c,d,e,c,d,e,f>
- <b,c,d,e>
- <a,c,d,b>
- <a,a,c,b,a,b,a,d,b,a,a,c,a,b>

Support: from 4/5 to **2/5** (informative, but not reflecting what happens within a trace)

Event-based support

$$\text{Constraint support (event-based)} = \frac{\text{\# events that satisfy the constraint activation and its target}}{\text{total \# events in the log}}$$

Response(a,b)

- <a,c,b,a>
- <c,d,e,c,d,e,f>
- <b,c,d,e>
- <a,c,d,b>
- <a,a,c,b,a,b,a,d,b,a,a,c,a,b>

Event-based support

$$\text{Constraint support (event-based)} = \frac{\text{\# events that satisfy the constraint activation and its target}}{\text{total \# events in the log}}$$

Response(a,b)

- <a,c,b,a>
- <c,d,e,c,d,e,f>
- <b,c,d,e>
- <a,c,d,b>
- <a,a,c,b,a,b,a,d,b,a,a,c,a,b>

Support: **9/33** (informative, but not reflecting trace satisfaction/violation)

Trace- and event-based confidence

Activation and target: solid basis to import the notion of confidence from association rule mining

$$\text{Constraint confidence (trace-based)} = \frac{\text{\# traces that satisfy the constraint and activate it}}{\text{\# traces that activate the constraint}}$$

$$\text{Constraint confidence (event-based)} = \frac{\text{\# events that satisfy the constraint activation and its target}}{\text{\# events that satisfy the constraint activation}}$$

Finally, discovery

[____,PMHandbook2022]

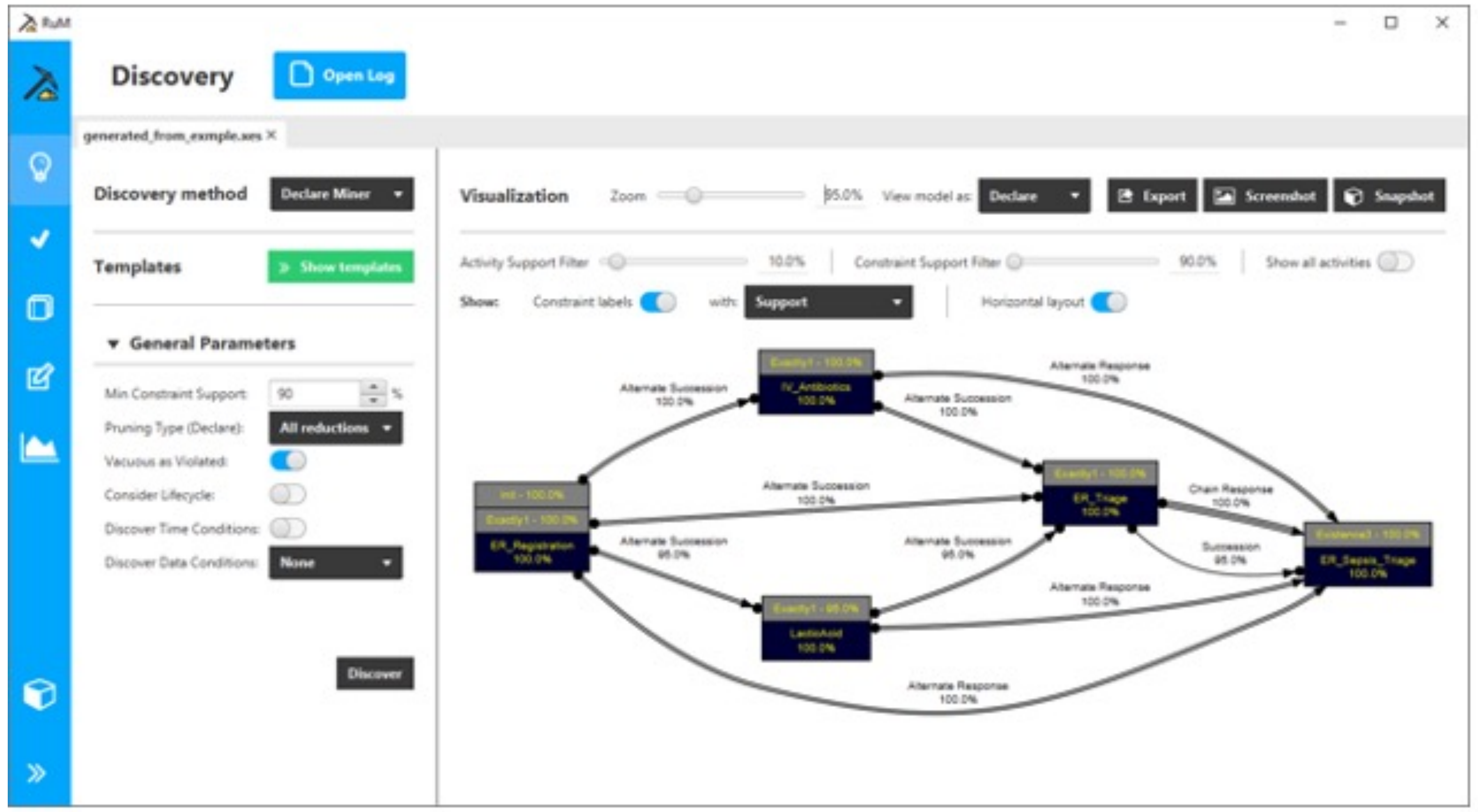
1. **Select templates** of interest
2. **Compute metrics** for corresponding constraints (grounded on log activities)
3. **Filter** based on minimum thresholds
4. **Redundant constraints**?
   - Keep **the most liberal** if **metrics are better** for it
   - Keep **the most restrictive** in case of **equal metrics**
5. **Incompatible constraints**?
   - Keep only the **one with better metrics**
6. **Further processing** to ensure consistency, minimality, …

automata

Tool support

RuM

https://svn.win.tue.nl/repos/prom/Packages/DeclareMiner/



Discovery method: Declare Miner

Min Constraint Support: 90 %

Pruning Type (Declare): All reductions

Vacuous as Violated: ●

Consider Lifecycle: ○

Discover Time Conditions: ○

Discover Data Conditions: None

Zoom: 95.0%

View model as: Declare

Activity Support Filter: 10.0%

Constraint Support Filter: 90.0%

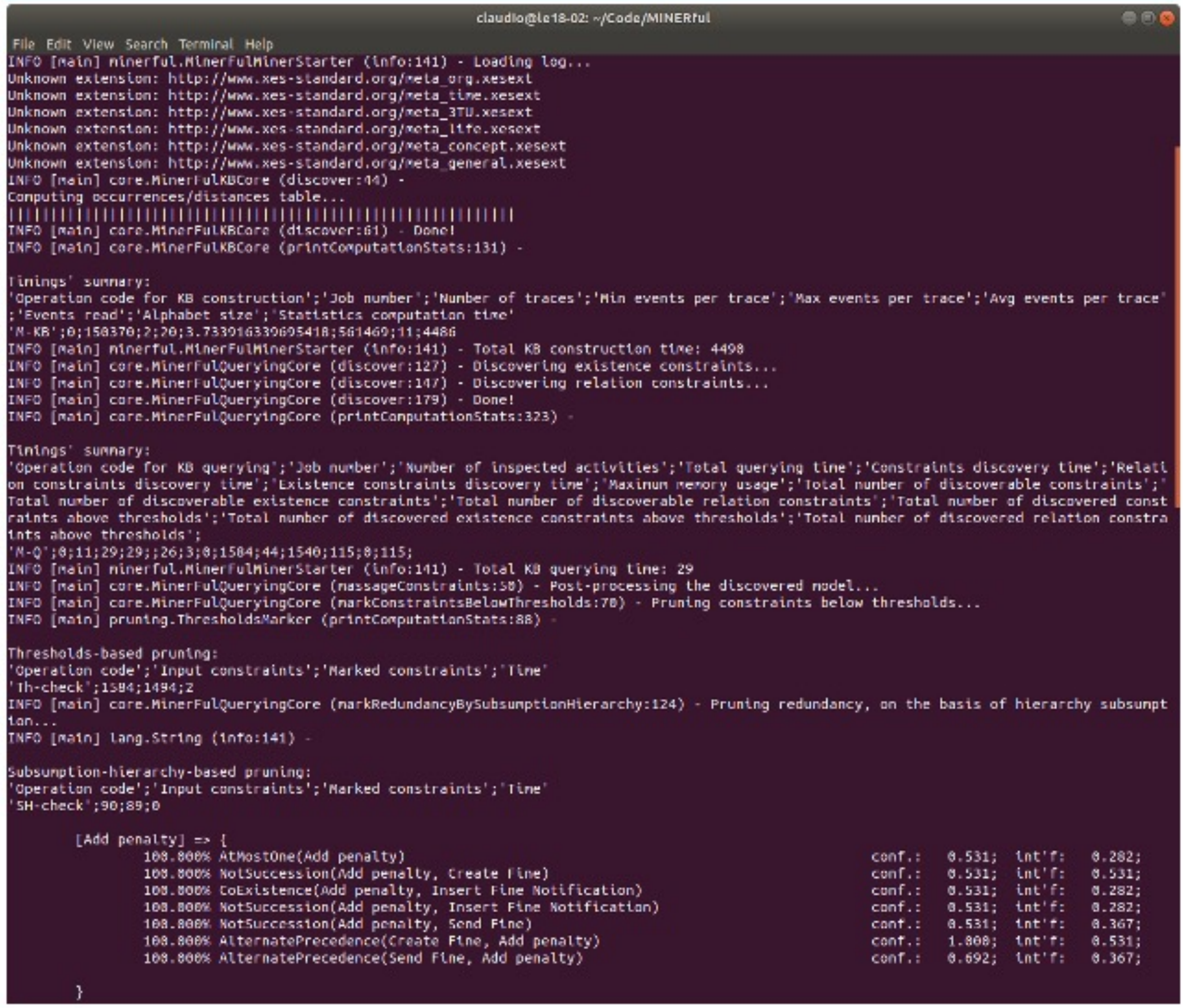
Show all activities: ○

Constraint labels: ● with: Support

Horizontal layout: ●

MINERful (command-line)

https://github.com/cdc08x/MINERful


claudio@le18-02: ~/Code/MINERful
File Edit View Search Terminal Help

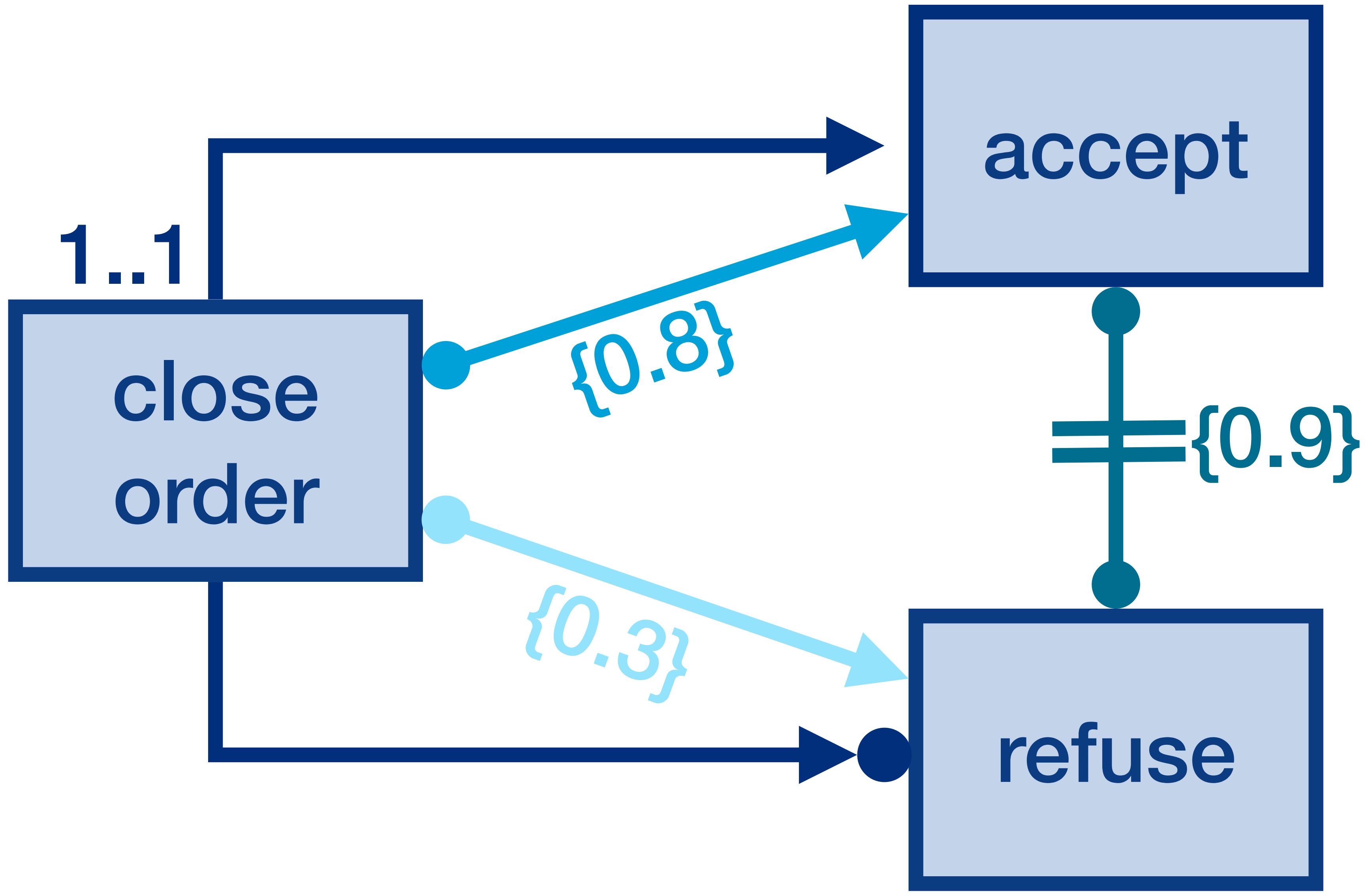
```
INFO [main] minerful.MinerFulMinerStarter (info:141) - Loading log...
Unknown extension: http://www.xes-standard.org/meta_org.xesext
Unknown extension: http://www.xes-standard.org/meta_time.xesext
Unknown extension: http://www.xes-standard.org/meta_3TU.xesext
Unknown extension: http://www.xes-standard.org/meta_life.xesext
Unknown extension: http://www.xes-standard.org/meta_concept.xesext
Unknown extension: http://www.xes-standard.org/meta_general.xesext
INFO [main] core.MinerFulKBCore (discover:44) -
Computing occurrences/distances table...
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
INFO [main] core.MinerFulKBCore (discover:61) - Done!
INFO [main] core.MinerFulKBCore (printComputationStats:131) -

Timings' summary:
'Operation code for KB construction';'Job number';'Number of traces';'Min events per trace';'Max events per trace';'Avg events per trace'
;'Events read';'Alphabet size';'Statistics computation time'
'M-KB';0;150370;2;20;3.733016330695418;561469;11;4486
INFO [main] minerful.MinerFulMinerStarter (info:141) - Total KB construction time: 4498
INFO [main] core.MinerFulQueryingCore (discover:127) - Discovering existence constraints...
INFO [main] core.MinerFulQueryingCore (discover:147) - Discovering relation constraints...
INFO [main] core.MinerFulQueryingCore (discover:179) - Done!
INFO [main] core.MinerFulQueryingCore (printComputationStats:323) -

Timings' summary:
'Operation code for KB querying';'Job number';'Number of inspected activities';'Total querying time';'Constraints discovery time';'Relati
on constraints discovery time';'Existence constraints discovery time';'Maximum memory usage';'Total number of discoverable constraints';'
Total number of discoverable existence constraints';'Total number of discoverable relation constraints';'Total number of discovered const
raints above thresholds';'Total number of discovered existence constraints above thresholds';'Total number of discovered relation constra
ints above thresholds';
'M-Q';0;11;29;29;;26;3;0;1584;44;1540;115;0;115;
INFO [main] minerful.MinerFulMinerStarter (info:141) - Total KB querying time: 29
INFO [main] core.MinerFulQueryingCore (massageConstraints:50) - Post-processing the discovered model...
INFO [main] core.MinerFulQueryingCore (markConstraintsBelowThresholds:70) - Pruning constraints below thresholds...
INFO [main] pruning.ThresholdsMarker (printComputationStats:88) -

Thresholds-based pruning:
'Operation code';'Input constraints';'Marked constraints';'Time'
'Th-check';1584;1494;2
INFO [main] core.MinerFulQueryingCore (markRedundancyBySubsumptionHierarchy:124) - Pruning redundancy, on the basis of hierarchy subsumpt
ion...
INFO [main] lang.String (info:141) -

Subsumption-hierarchy-based pruning:
'Operation code';'Input constraints';'Marked constraints';'Time'
'SH-check';90;89;0

        [Add penalty] => {
                100.000% AtMostOne(Add penalty)                                                     conf.:   0.531;  int'f:   0.282;
                100.000% NotSuccession(Add penalty, Create Fine)                                    conf.:   0.531;  int'f:   0.531;
                100.000% CoExistence(Add penalty, Insert Fine Notification)                         conf.:   0.531;  int'f:   0.282;
                100.000% NotSuccession(Add penalty, Insert Fine Notification)                       conf.:   0.531;  int'f:   0.282;
                100.000% NotSuccession(Add penalty, Send Fine)                                      conf.:   0.531;  int'f:   0.367;
                100.000% AlternatePrecedence(Create Fine, Add penalty)                              conf.:   1.000;  int'f:   0.531;
                100.000% AlternatePrecedence(Send Fine, Add penalty)                                conf.:   0.692;  int'f:   0.367;

        }
```

5 exciting research lines

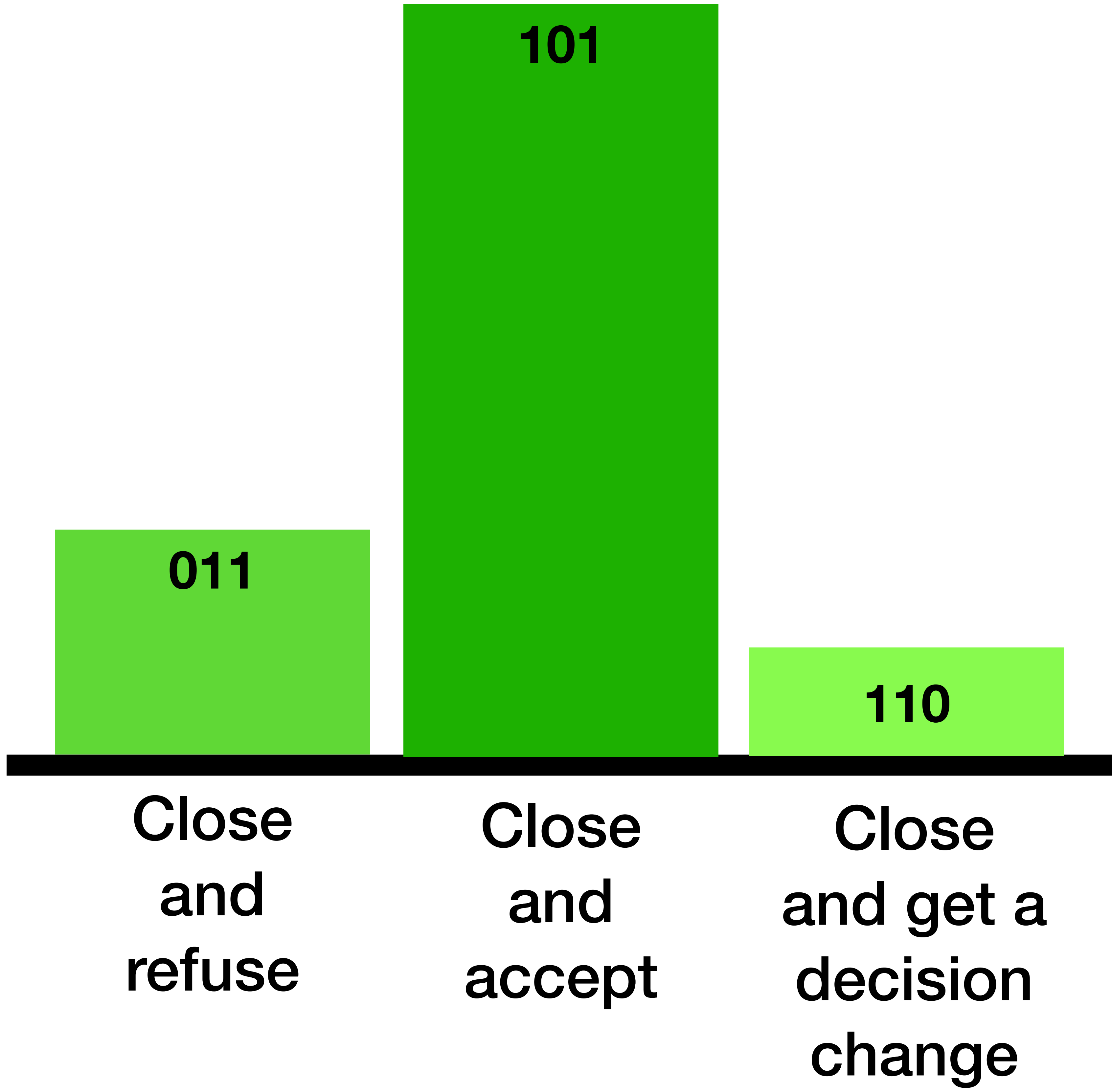
1. Dealing with uncertainty

Probabilistic LTLf and applications

[____,AAAI2020] [____,BPM2020] [____,InfSys2022]



scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0



2. Dealing with data

Monitoring/enactment with numerical data variables

[____,AAAI2022] [____,AAAI2023]

LTLf over numerical variables with arithmetic conditions

- **Undecidability** around the corner

Identification of **decidable fragments** tuning **condition language** and/or **variable interaction**

- Semantic notion of **finite summary**, yielding **decidability**
- Concrete instantiations **reproduce and generalise known classes**

Lifting of **automata-based techniques** using **SMT reasoners** to deal with conditions

Monitoring/enactment with numerical data variables

[____,AAAI2022] [____,AAAI2023]

LTLf over numerical variables with arithmetic conditions

- **Undecidability** around the corner

Identification of **de** ning **condition language** and/or **v**

- Semantic notion ding **decidability**
- Concrete instanti **generalise known classes**

Lifting of **automata-** g **SMT reasoners** to deal with conditions

Extended to variables pointing to complex data structures

Monitoring/enactment with numerical data variables

[____ AI2022] [____,AAAI2023]

LTL al variables with arithmetic conditions

- he corner

ning condition

- Conc classes

Lifting of automata- deal with conditions

See Sarah Winkler's presentation later!

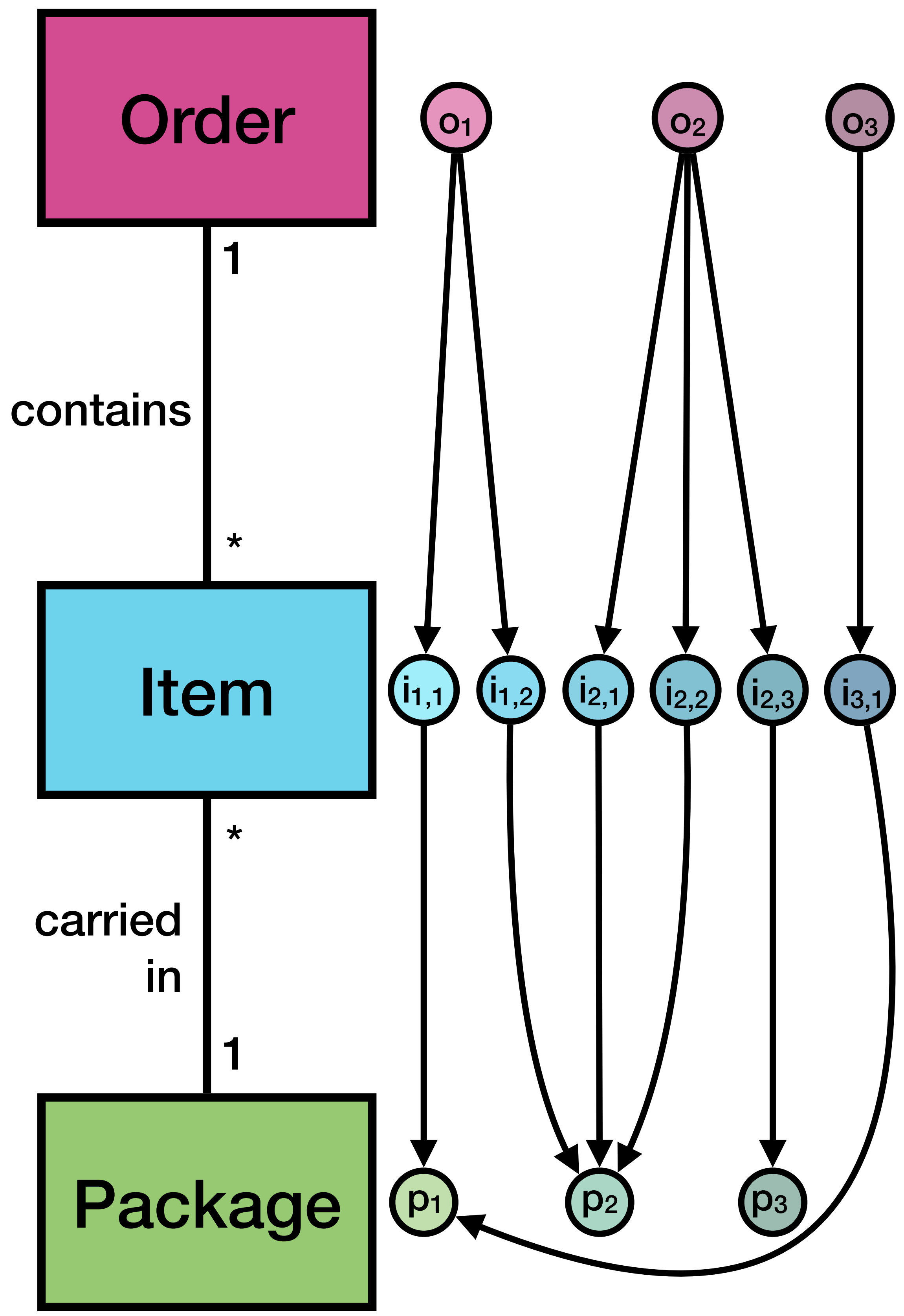
Extended to variables pointing to complex data structures

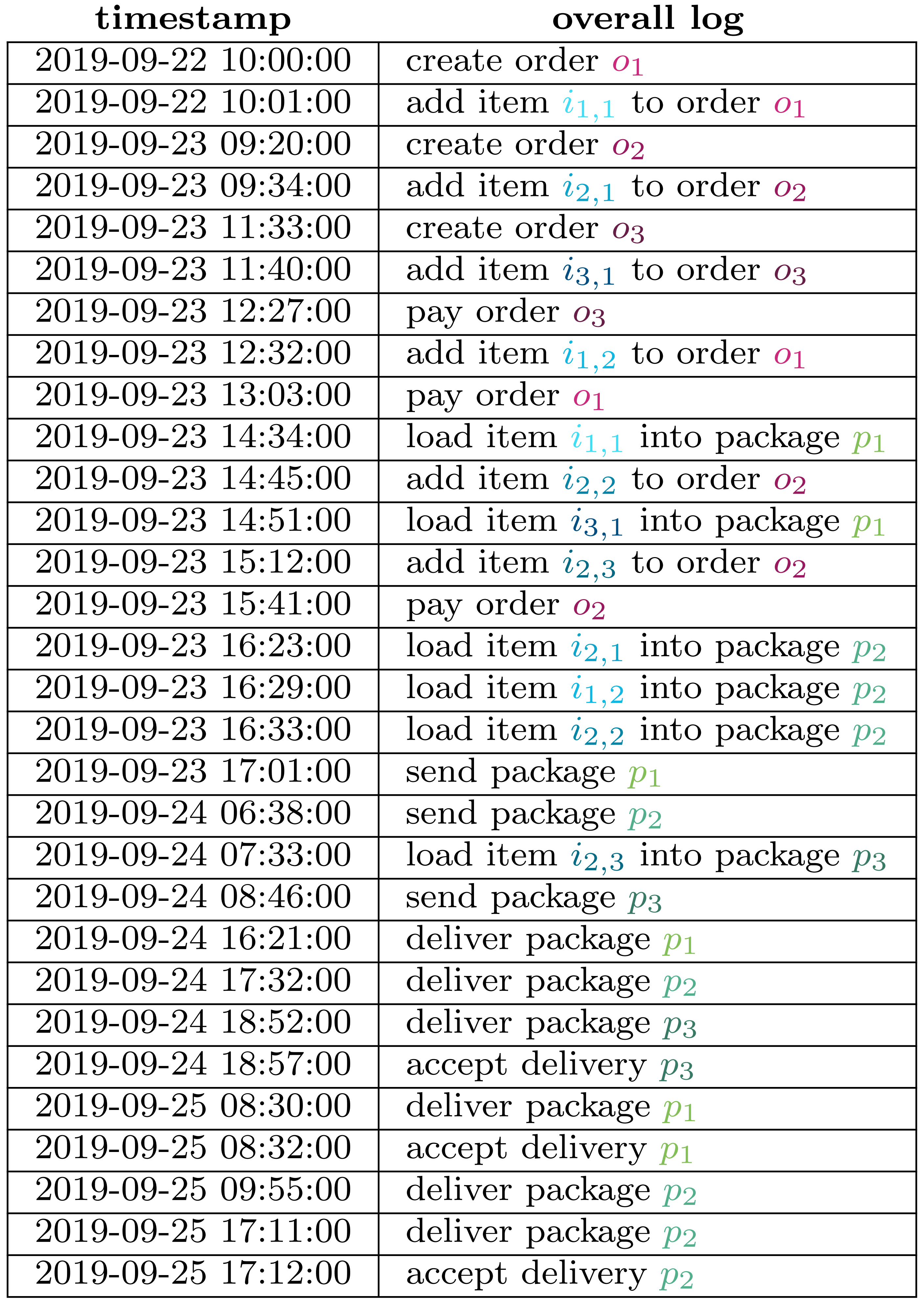
Also Nicola Gigante on related topics

(Wed 9:00am)

3. Dealing with multiple objects

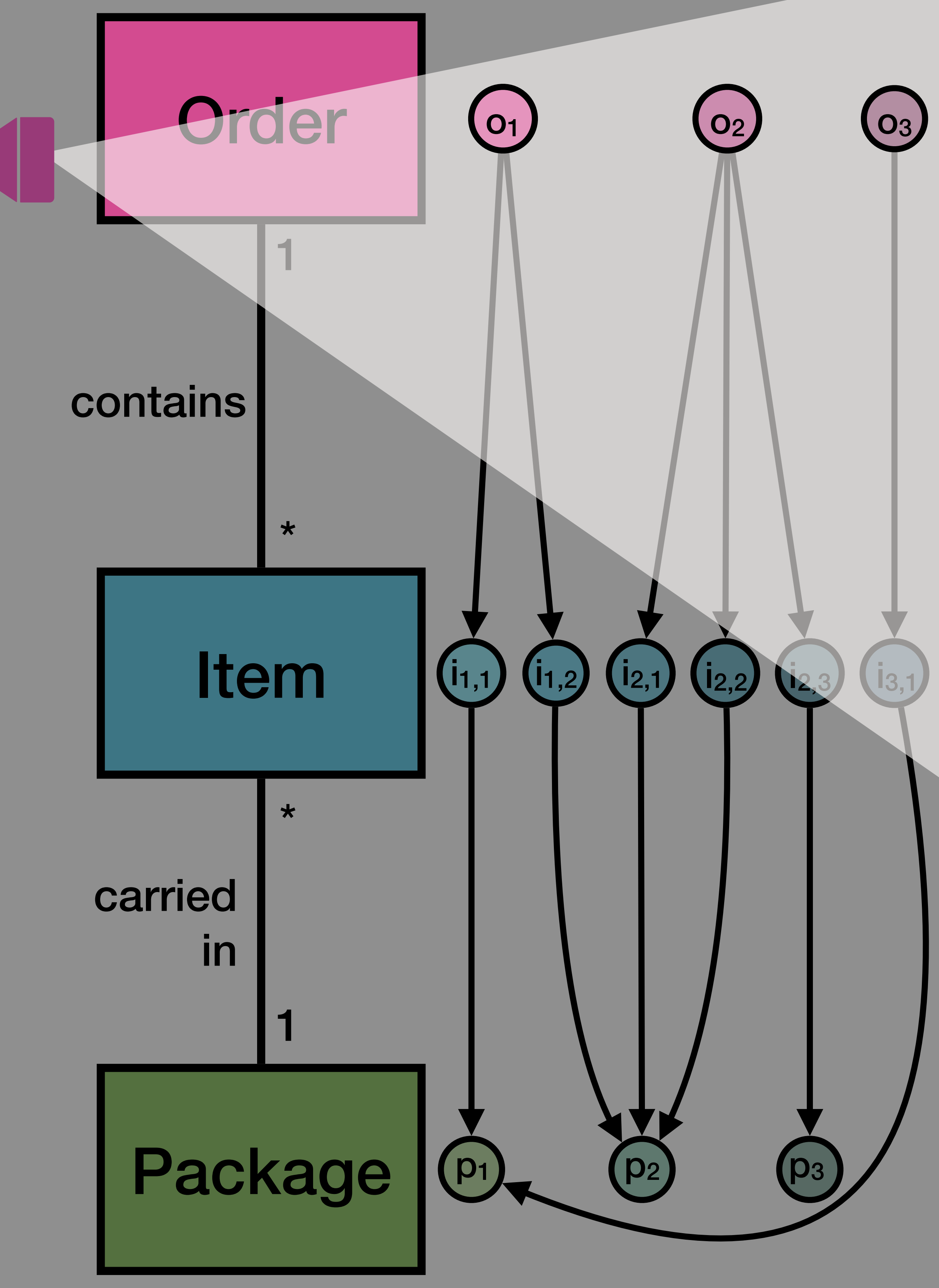
Processes are not flat





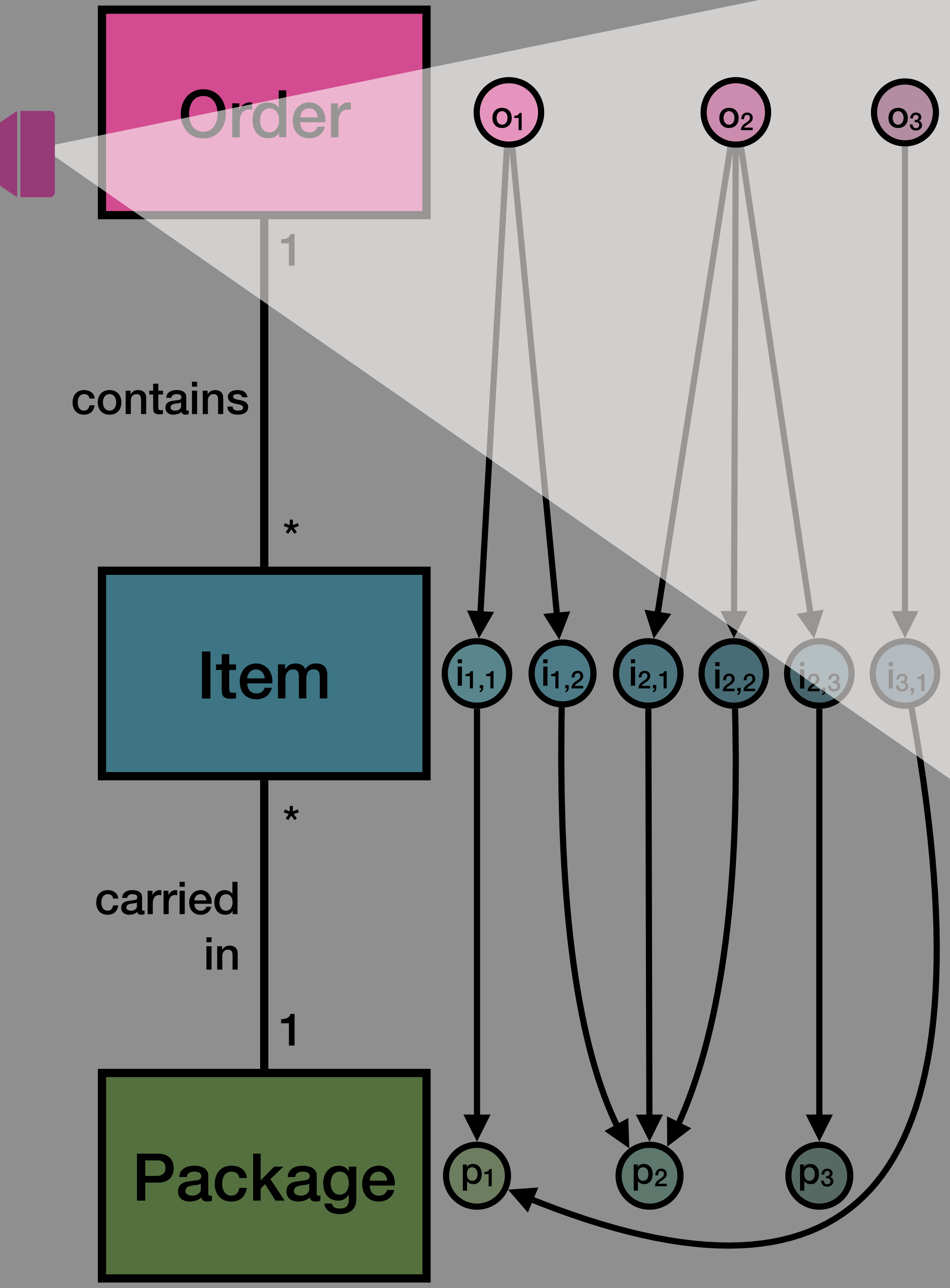
timestamp	overall log
2019-09-22 10:00:00	create order o_1
2019-09-22 10:01:00	add item $i_{1,1}$ to order o_1
2019-09-23 09:20:00	create order o_2
2019-09-23 09:34:00	add item $i_{2,1}$ to order o_2
2019-09-23 11:33:00	create order o_3
2019-09-23 11:40:00	add item $i_{3,1}$ to order o_3
2019-09-23 12:27:00	pay order o_3
2019-09-23 12:32:00	add item $i_{1,2}$ to order o_1
2019-09-23 13:03:00	pay order o_1
2019-09-23 14:34:00	load item $i_{1,1}$ into package p_1
2019-09-23 14:45:00	add item $i_{2,2}$ to order o_2
2019-09-23 14:51:00	load item $i_{3,1}$ into package p_1
2019-09-23 15:12:00	add item $i_{2,3}$ to order o_2
2019-09-23 15:41:00	pay order o_2
2019-09-23 16:23:00	load item $i_{2,1}$ into package p_2
2019-09-23 16:29:00	load item $i_{1,2}$ into package p_2
2019-09-23 16:33:00	load item $i_{2,2}$ into package p_2
2019-09-23 17:01:00	send package p_1
2019-09-24 06:38:00	send package p_2
2019-09-24 07:33:00	load item $i_{2,3}$ into package p_3
2019-09-24 08:46:00	send package p_3
2019-09-24 16:21:00	deliver package p_1
2019-09-24 17:32:00	deliver package p_2
2019-09-24 18:52:00	deliver package p_3
2019-09-24 18:57:00	accept delivery p_3
2019-09-25 08:30:00	deliver package p_1
2019-09-25 08:32:00	accept delivery p_1
2019-09-25 09:55:00	deliver package p_2
2019-09-25 17:11:00	deliver package p_2
2019-09-25 17:12:00	accept delivery p_2

Processes are not flat



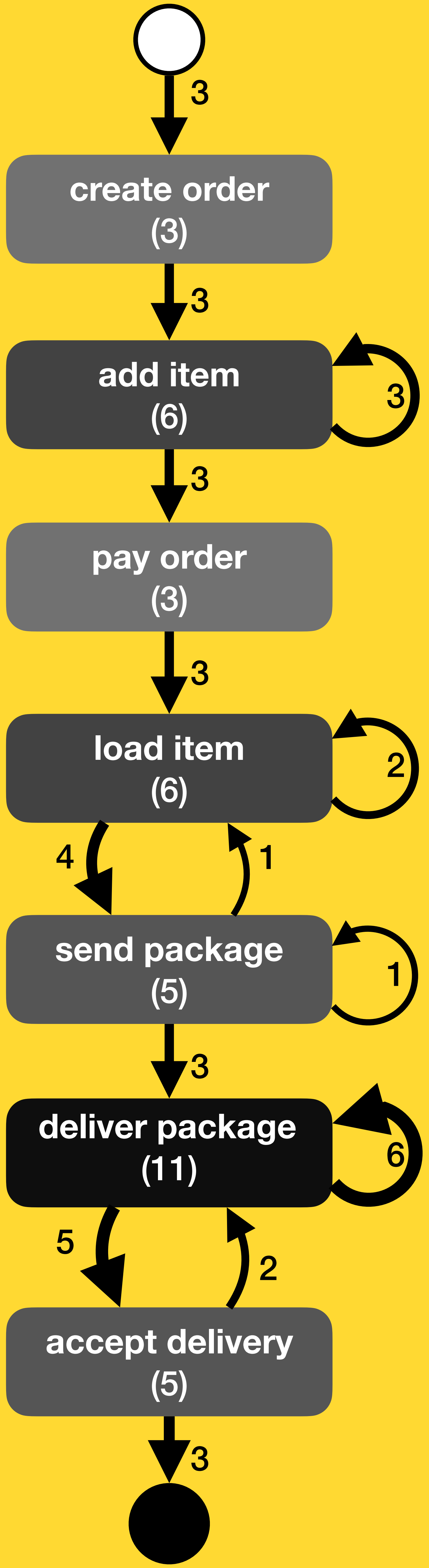
			event log for orders	
timestamp	overall log	order o_1	order o_2	order o_3
2019-09-22 10:00:00	create order o_1	create order		
2019-09-22 10:01:00	add item $i_{1,1}$ to order o_1	add item		
2019-09-23 09:20:00	create order o_2		create order	
2019-09-23 09:34:00	add item $i_{2,1}$ to order o_2		add item	
2019-09-23 11:33:00	create order o_3			create order
2019-09-23 11:40:00	add item $i_{3,1}$ to order o_3			add item
2019-09-23 12:27:00	pay order o_3			pay order
2019-09-23 12:32:00	add item $i_{1,2}$ to order o_1	add item		
2019-09-23 13:03:00	pay order o_1	pay order		
2019-09-23 14:34:00	load item $i_{1,1}$ into package p_1	load item		
2019-09-23 14:45:00	add item $i_{2,2}$ to order o_2		add item	
2019-09-23 14:51:00	load item $i_{3,1}$ into package p_1			load item
2019-09-23 15:12:00	add item $i_{2,3}$ to order o_2		add item	
2019-09-23 15:41:00	pay order o_2		pay order	
2019-09-23 16:23:00	load item $i_{2,1}$ into package p_2		load item	
2019-09-23 16:29:00	load item $i_{1,2}$ into package p_2	load item		
2019-09-23 16:33:00	load item $i_{2,2}$ into package p_2		load item	
2019-09-23 17:01:00	send package p_1	send package		send package
2019-09-24 06:38:00	send package p_2	send package	send package	
2019-09-24 07:33:00	load item $i_{2,3}$ into package p_3			load item
2019-09-24 08:46:00	send package p_3		send package	
2019-09-24 16:21:00	deliver package p_1	deliver package		deliver package
2019-09-24 17:32:00	deliver package p_2	deliver package	deliver package	
2019-09-24 18:52:00	deliver package p_3		deliver package	
2019-09-24 18:57:00	accept delivery p_3		accept delivery	
2019-09-25 08:30:00	deliver package p_1	deliver package		deliver package
2019-09-25 08:32:00	accept delivery p_1	accept delivery		accept delivery
2019-09-25 09:55:00	deliver package p_2	deliver package	deliver package	
2019-09-25 17:11:00	deliver package p_2	deliver package	deliver package	
2019-09-25 17:12:00	accept delivery p_2	accept delivery	accept delivery	

Processes are not flat



timestamp	overall log
2019-09-22 10:00:00	create order o_1
2019-09-22 10:01:00	add item $i_{1,1}$ to order o_1
2019-09-23 09:20:00	create order o_2
2019-09-23 09:34:00	add item $i_{2,1}$ to order o_2
2019-09-23 11:33:00	create order o_3
2019-09-23 11:40:00	add item $i_{3,1}$ to order o_3
2019-09-23 12:27:00	pay order o_3
2019-09-23 12:32:00	add item $i_{1,2}$ to order o_1
2019-09-23 13:03:00	pay order o_1
2019-09-23 14:34:00	load item $i_{1,1}$ into package p_1
2019-09-23 14:45:00	add item $i_{2,2}$ to order o_2
2019-09-23 14:51:00	load item $i_{3,1}$ into package p_1
2019-09-23 15:12:00	add item $i_{2,3}$ to order o_2
2019-09-23 15:41:00	pay order o_2
2019-09-23 16:23:00	load item $i_{2,1}$ into package p_2
2019-09-23 16:29:00	load item $i_{1,2}$ into package p_2
2019-09-23 16:33:00	load item $i_{2,2}$ into package p_2
2019-09-23 17:01:00	send package p_1
2019-09-24 06:38:00	send package p_2
2019-09-24 07:33:00	load item $i_{2,3}$ into package p_3
2019-09-24 08:46:00	send package p_3
2019-09-24 16:21:00	deliver package p_1
2019-09-24 17:32:00	deliver package p_2
2019-09-24 18:52:00	deliver package p_3
2019-09-24 18:57:00	accept delivery p_3
2019-09-25 08:30:00	deliver package p_1
2019-09-25 08:32:00	accept delivery p_1
2019-09-25 09:55:00	deliver package p_2
2019-09-25 17:11:00	deliver package p_2
2019-09-25 17:12:00	accept delivery p_2

	event log for orders	
order o_1	order o_2	order o_3
create order		
add item		
	create order	
	add item	
		create order
		add item
		pay order
add item		
pay order		
load item		
	add item	
		load item
	add item	
	pay order	
	load item	
load item		
	load item	
send package		send package
send package	send package	
		load item
	send package	
deliver package		deliver package
deliver package	deliver package	
	deliver package	
	accept delivery	
deliver package		deliver package
accept delivery		accept delivery
deliver package	deliver package	
deliver package	deliver package	
accept delivery	accept delivery	



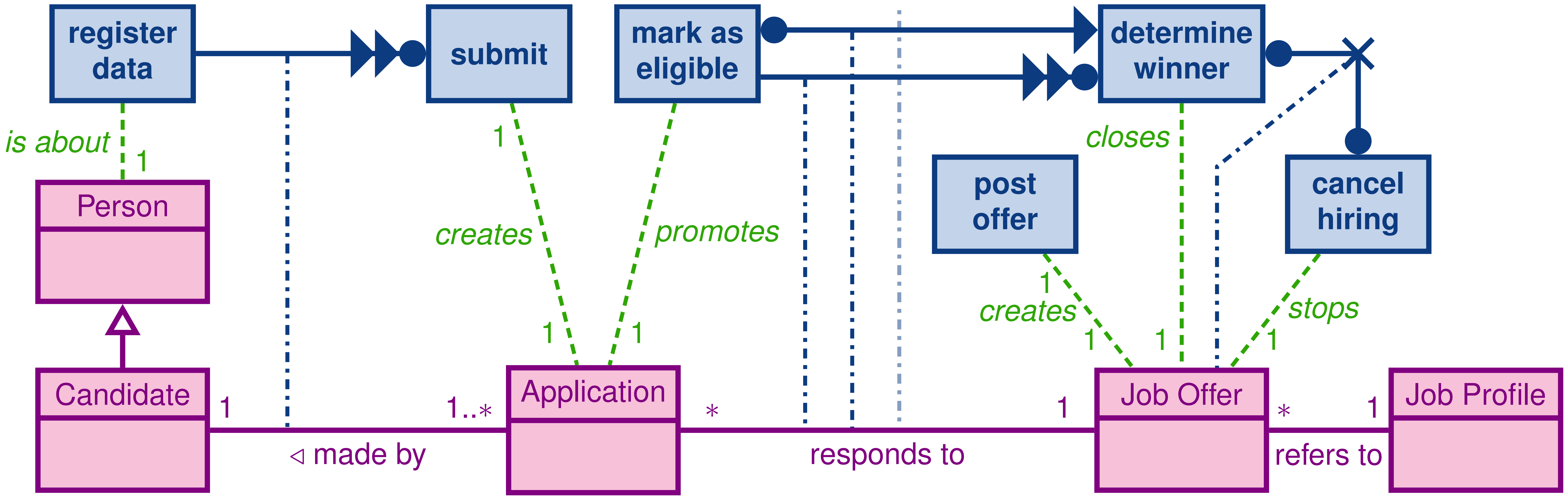
Object-centric behavioral constraints

[____,DL2015] [____,BPM2019]

(FO-)LTLf constraints **co-referring** through **objects** and **relations**

- Models: **temporal knowledge graphs**
- **Undecidability** around the corner, **decidable** for **Declare+ALCQI**

Challenge in **balancing "open" vs "closed" semantics**

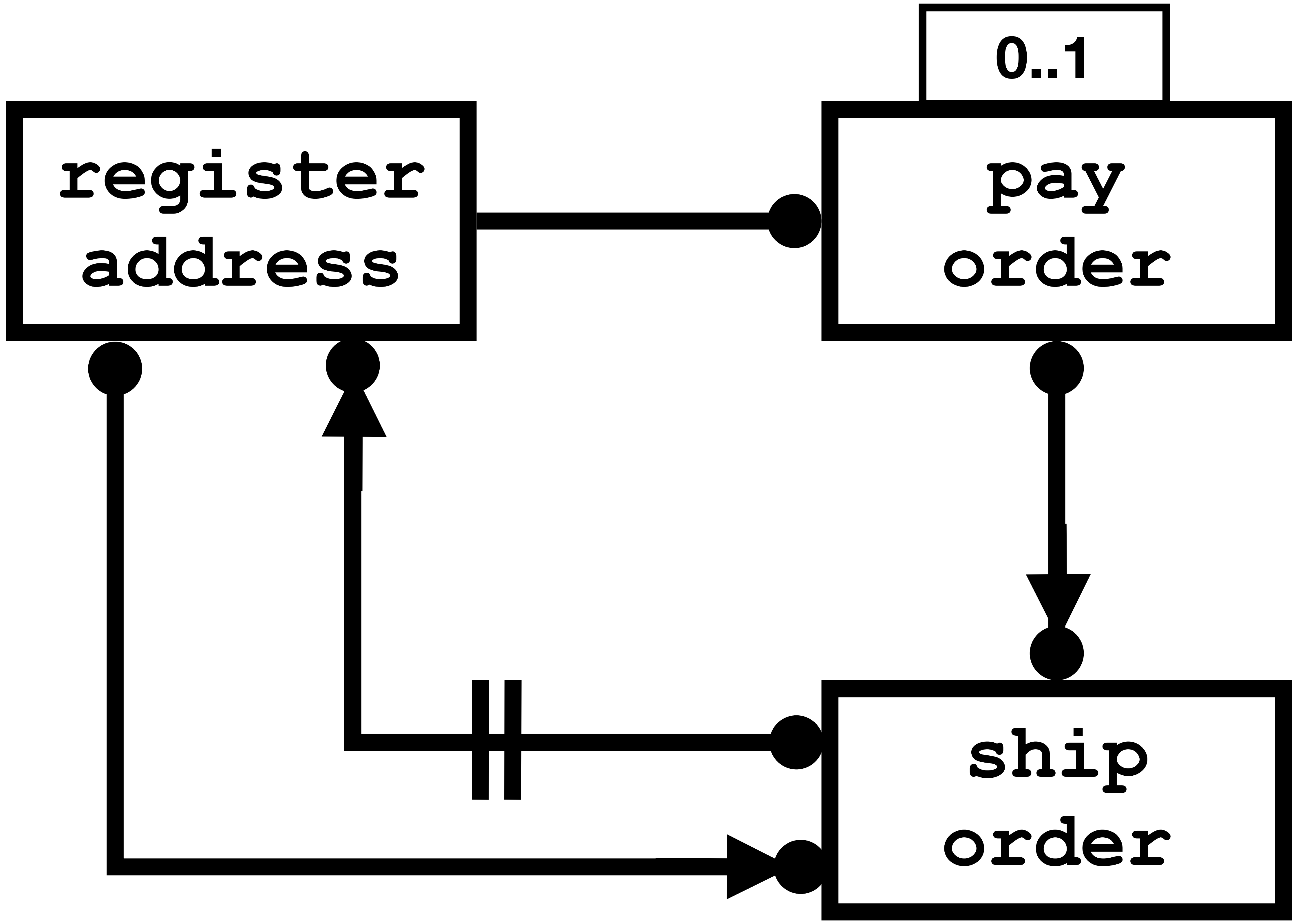


4. Multi-party declarative processes

Collaborative Declare

Tasks/constraints distributed to controller and environment

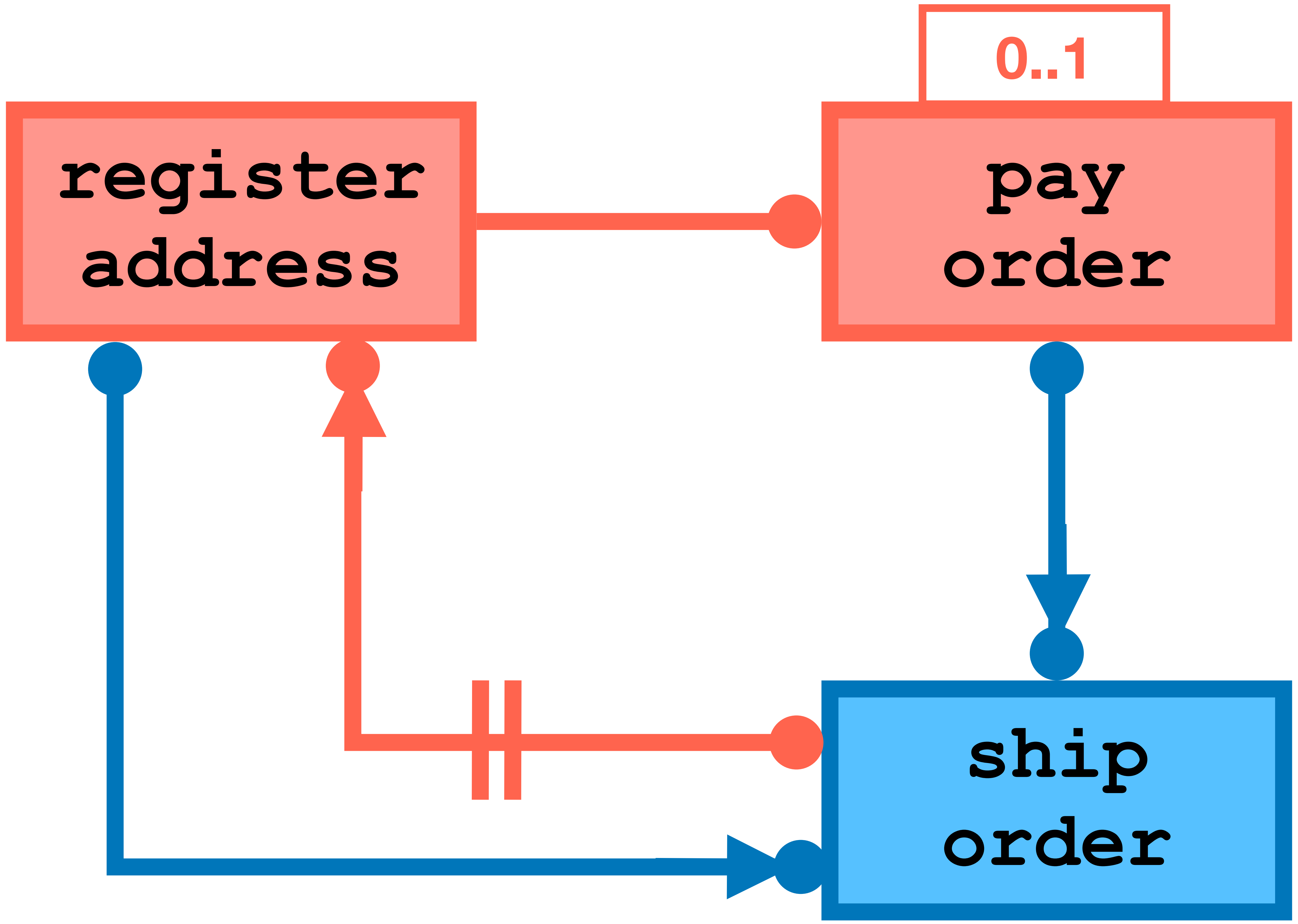
- From enactment to assume-guarantee realizability/synthesis



Collaborative Declare

Tasks/**constraints** distributed to **controller** and **environment**

- From enactment to **assume-guarantee realizability/synthesis**



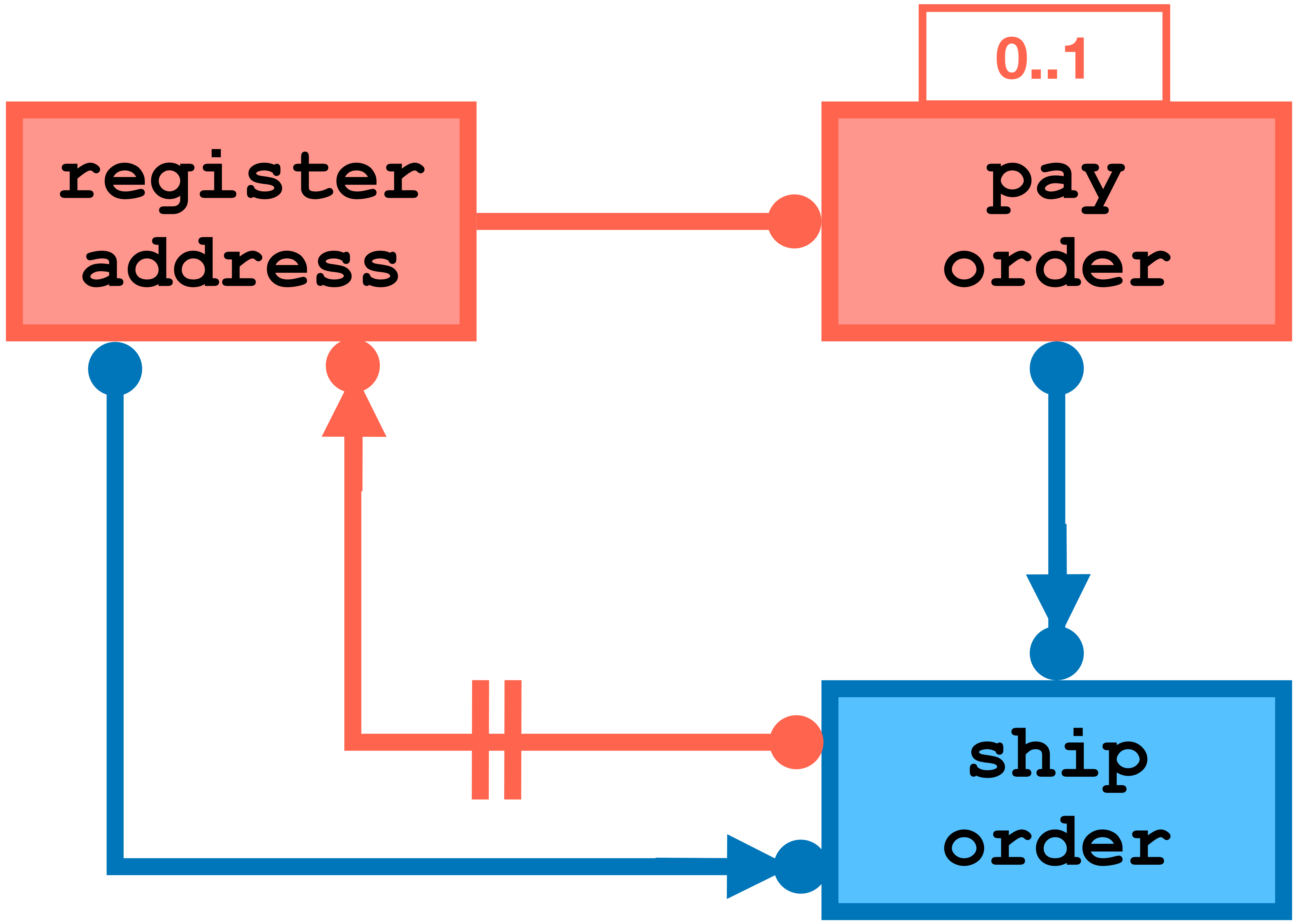
Collaborative Declare

Tasks/**constraints** distributed to **controller** and **environment**

- From enactment to **assume-guarantee realizability/synthesis**

Realizability in **single-ExpTime**

- Declare yields **poly-size pastification**
- **Linear encoding** into **symbolic DFAs**



Collaborative Declare

Tasks/**constraints** distributed to c ler and **environment**

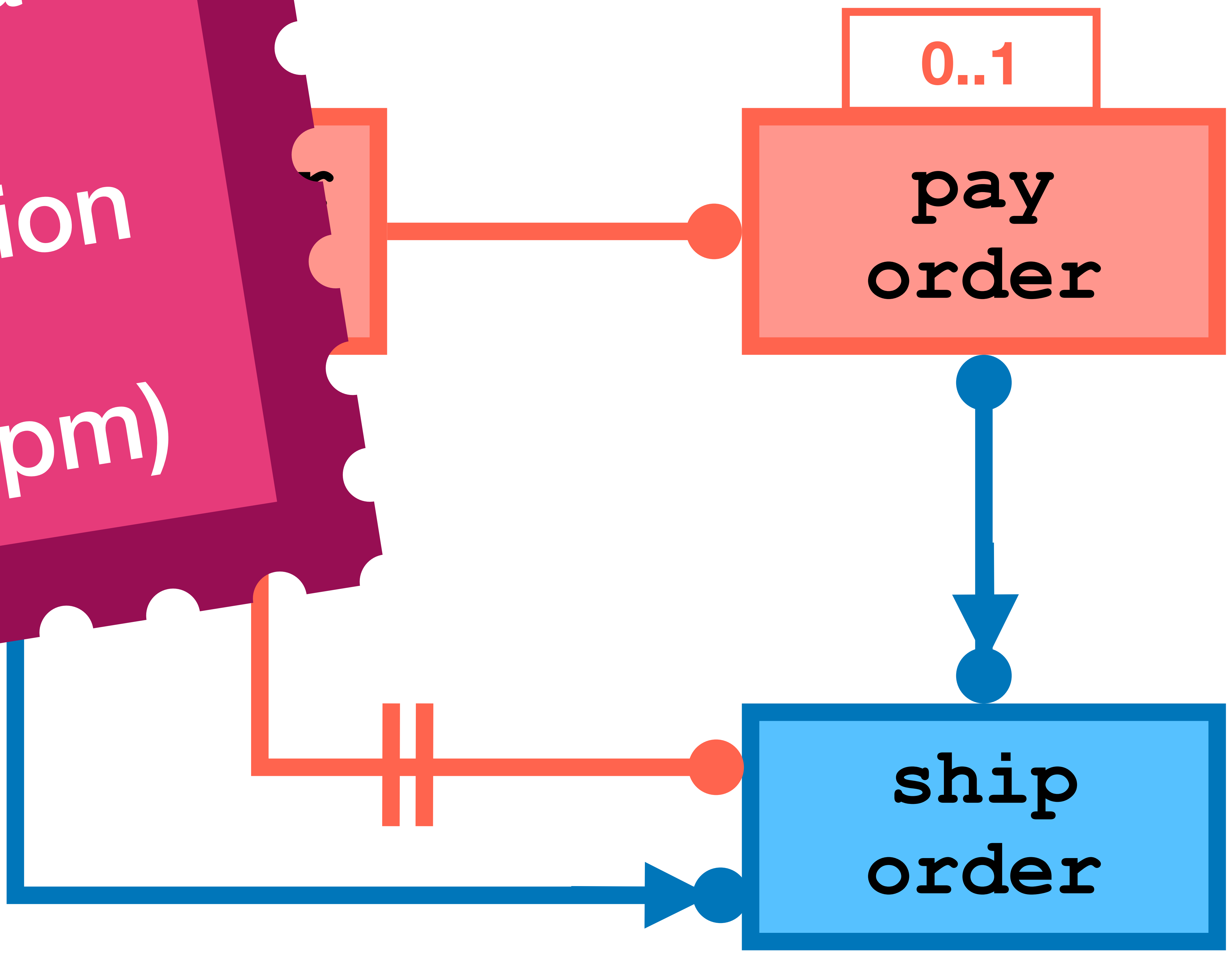
- From enactment t **realizability/synthesis**

Realizability in **singl**

- Declare yields **poly-size pastifica**
- **Linear encoding** in **symbolic DFAs**

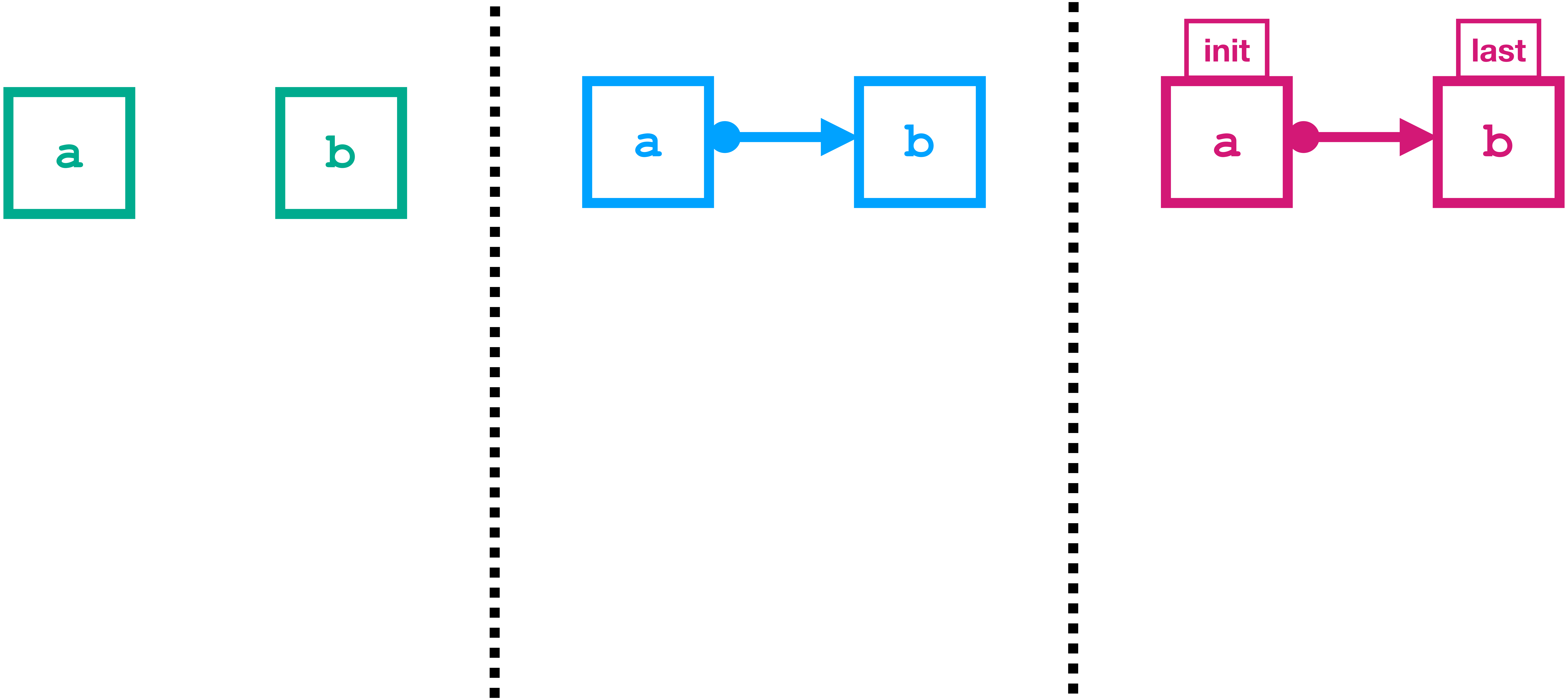
See Luca Geatti's presentation

(Tue 4:50pm)



5. Measuring flexibility

How flexible?

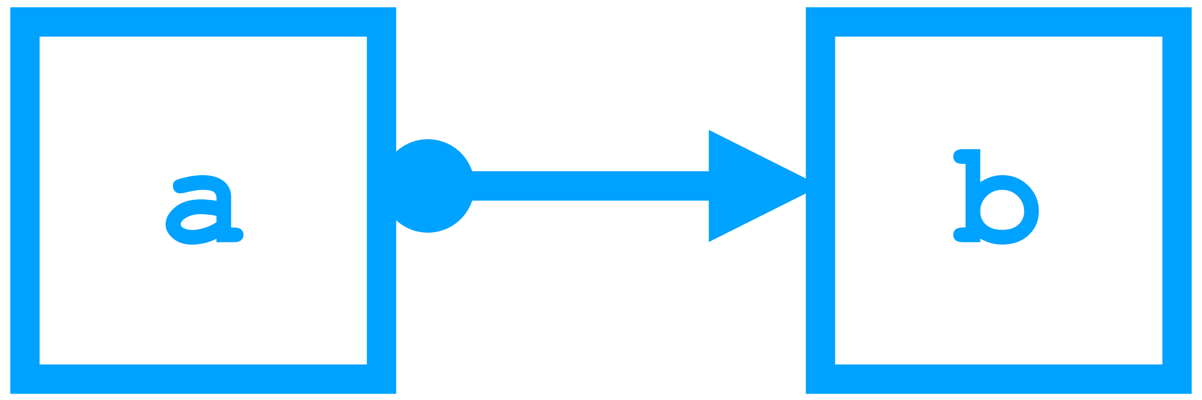


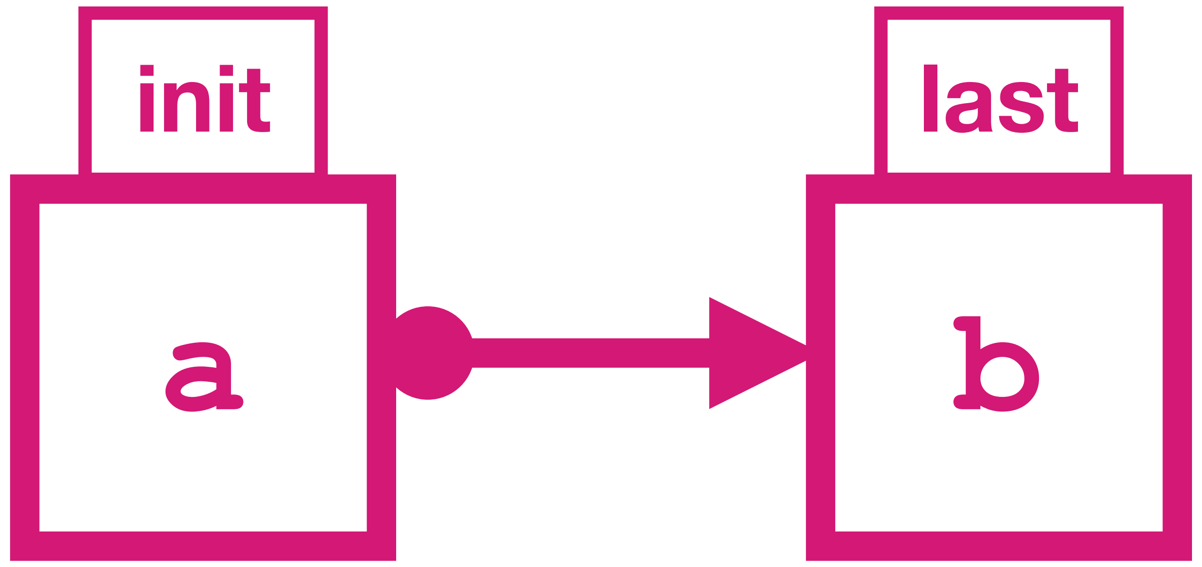
How flexible?

infinitely many traces…

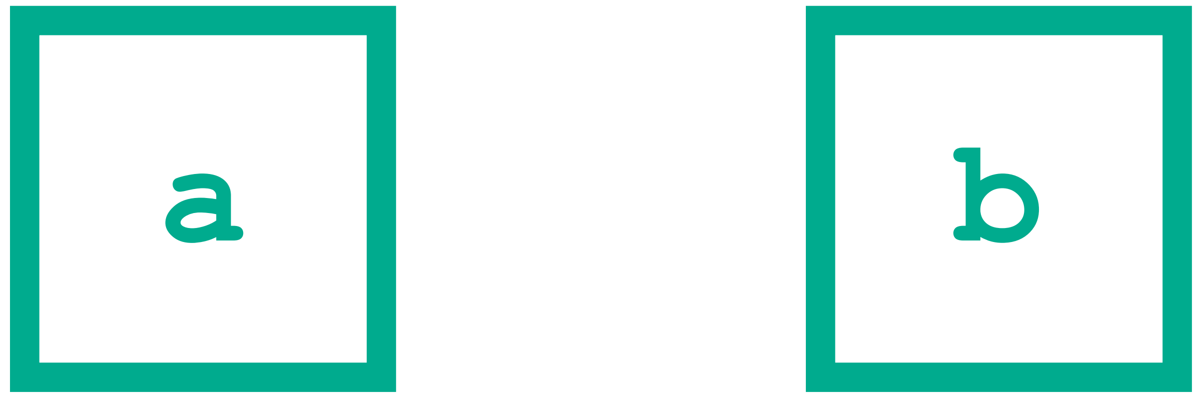
… actually all! (Σ*)

flexibility = 1





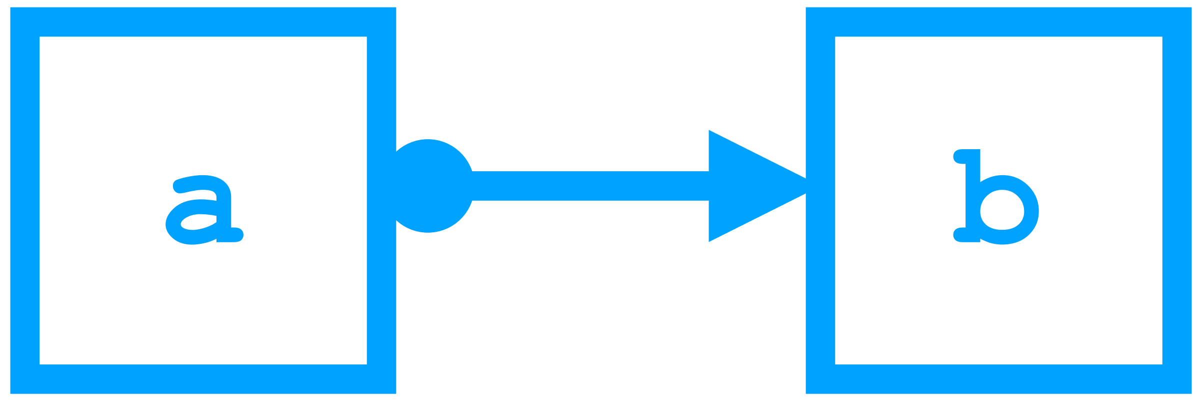
How flexible?

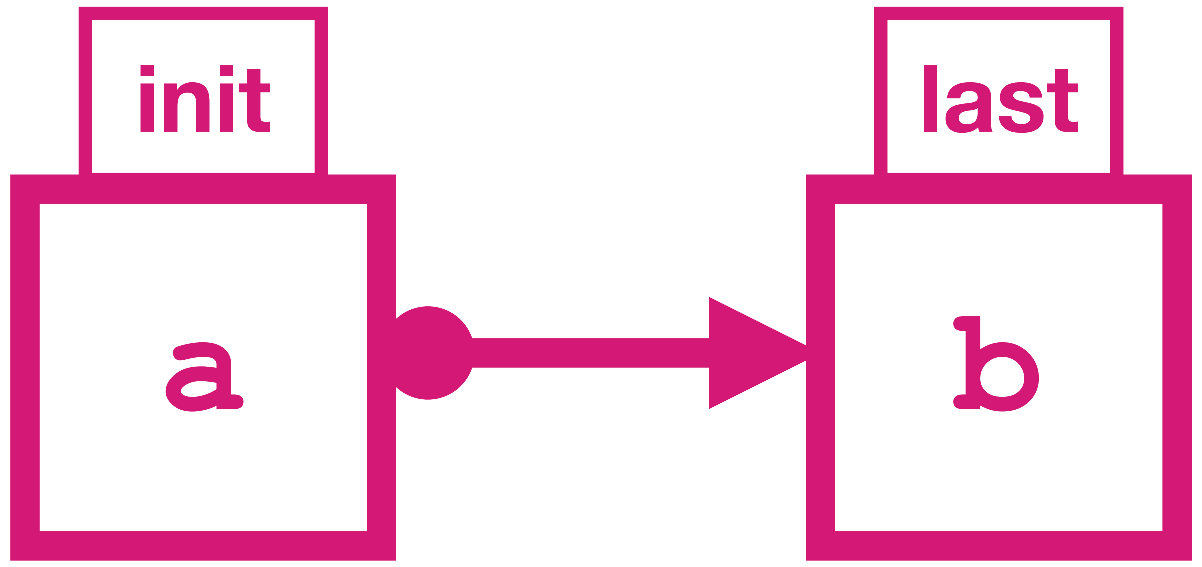


infinitely many traces…

… actually all! (Σ*)

flexibility = 1



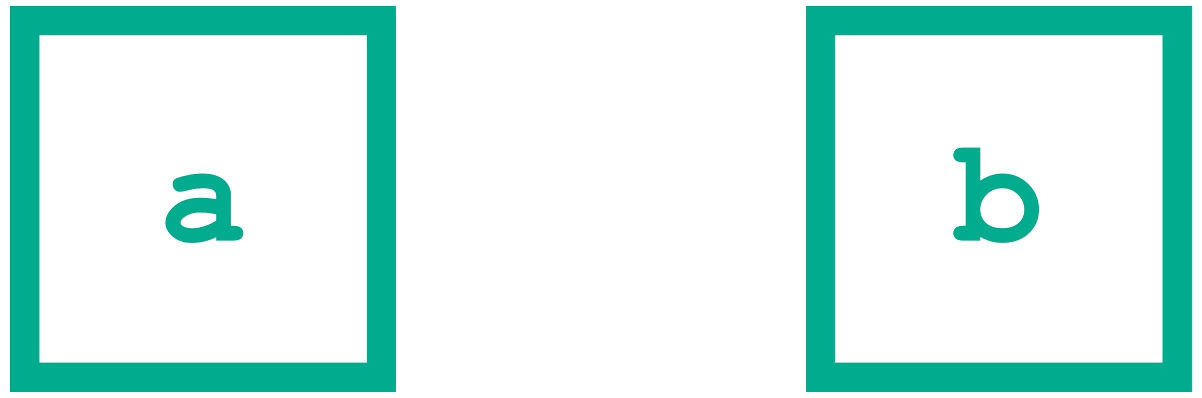


only one trace…

…do “a” and then do “b”

flexibility = 0

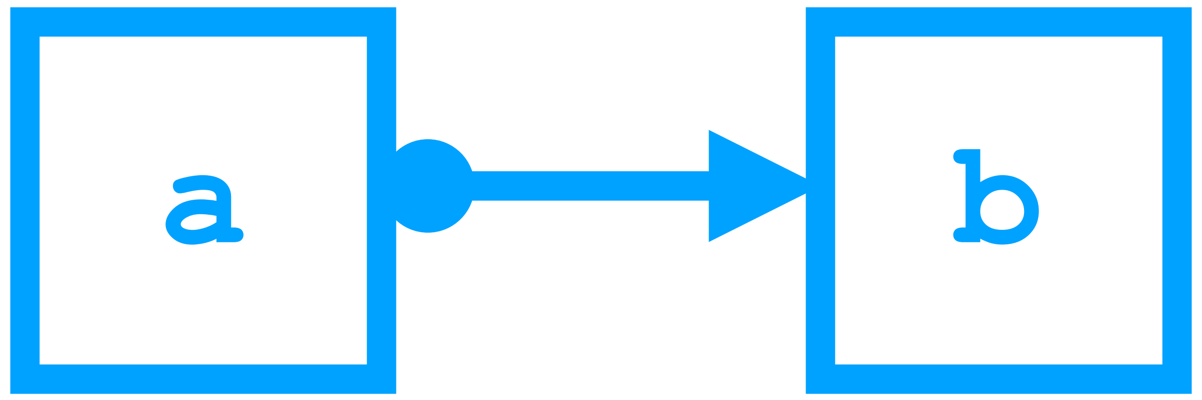
How flexible?



infinitely many traces…

… actually all! (Σ^*)

flexibility = 1

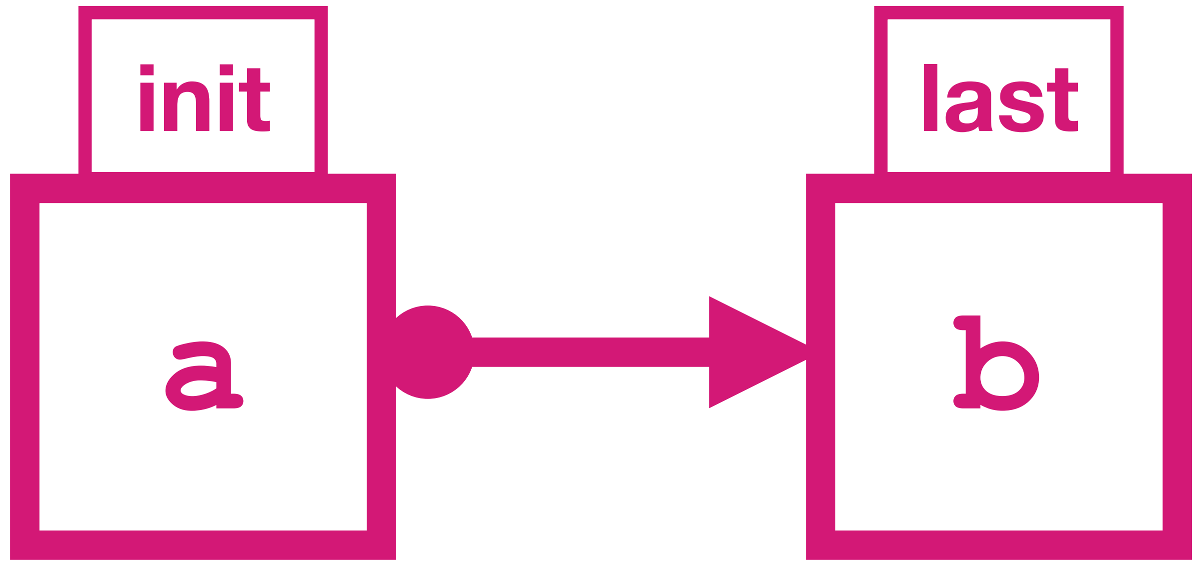


infinitely many traces

…finishing with “a”

0 < flexibility < 1

= 0.5



only one trace…

…do “a” and then do “b”

flexibility = 0

Flexibility as “density of traces”

Given a task alphabet Σ and a regular language $\mathscr{L}$ over Σ…

$$\text{flex}(\mathscr{L}) = \qquad \frac{\qquad\qquad}{\qquad\qquad}$$

Flexibility as “density of traces”

Given a task alphabet Σ and a regular language $\mathscr{L}$ over Σ…

number of traces of length up to n accepted by $\mathscr{L}$

$$\mathrm{flex}(\mathscr{L}) = \frac{W_{\leq n}(\mathscr{L})}{\phantom{W_{\leq n}(\mathscr{L})}}$$

Flexibility as “density of traces”

Given a task alphabet Σ and a regular language $\mathscr{L}$ over Σ…

number of traces of length up to n accepted by $\mathscr{L}$

$$\text{flex}(\mathscr{L}) = \frac{W_{\leq n}(\mathscr{L})}{W_{\leq n}(\Sigma^*)}$$

maximally flexible behavior

Flexibility as “density of traces”

Given a task alphabet Σ and a regular language $\mathscr{L}$ over Σ…

number of traces of length up to n accepted by $\mathscr{L}$

$$\text{flex}(\mathscr{L}) = lim_{n\to\infty}\frac{W_{\leq n}(\mathscr{L})}{W_{\leq n}(\Sigma^*)}$$

maximally flexible behavior

Flexibility as “density of traces”

Given a task alphabet Σ and a regular language $\mathscr{L}$ over Σ…

number of traces of length up to n accepted by $\mathscr{L}$

$$\mathrm{flex}(\mathscr{L}) = lim_{n \to \infty} \frac{W_{\leq n}(\mathscr{L})}{W_{\leq n}(\Sigma^*)}$$

maximally flexible behavior

For **Declare** constraints and their boolean combinations:
the limit always exists

When the limit exists: computable using techniques based on **topological entropy of DFAs**!

Wrapping up

Conclusions

Managing and mining **flexible processes** is an **open challenge**

Declarative specifications tackle **flexibility by design**

Solid foundations based on **LTLf**

Automata at the core of **all analysis and support tasks**
(no ad-hoc algorithms!)

Fascinating research **synergically combining** and
jointly advancing **AI** and **information systems**

Thanks to Wil van der Aalst, Anti Alman, Alessandro Artale, Federico Chesani, Giuseppe De Giacomo, Riccardo De Masellis, Claudio Di Ciccio, Marlon Dumas, Dirk Fahland, Paolo Felli, Alessandro Gianola, Alisa Kovtunova, Fabrizio Maggi, Andrea Marrella, Paola Mello, Jan Mendling, Fabio Patrizi, Rafael Penaloza, Maja Pesic, Andrey Rivkin, Michael Westergaard, Sarah Winkler

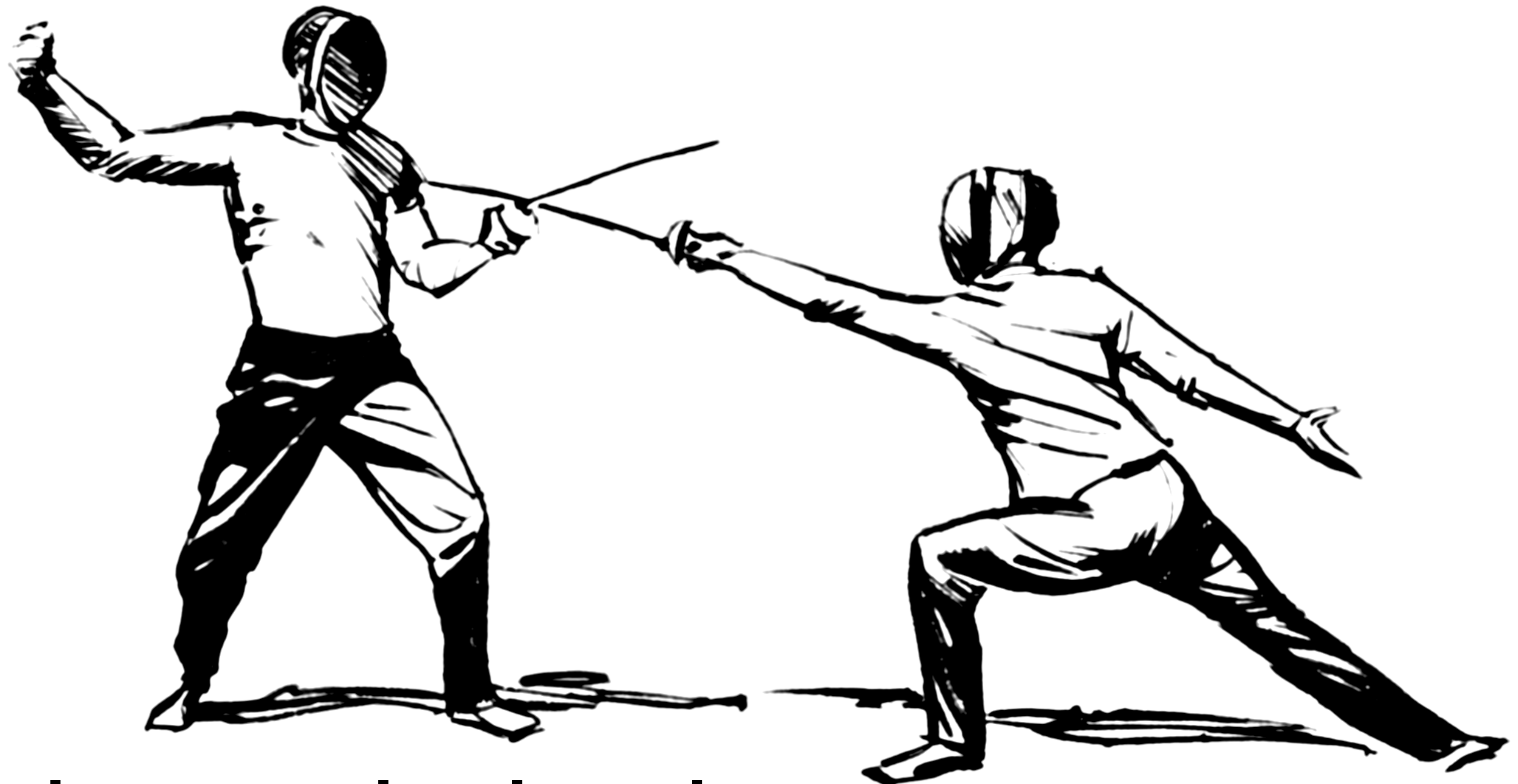
Thank you!

Marco Montali

montali@inf.unibz.it

Challenging Declare

Frequencies and uncertainty



- **Best practices**: constraints that must hold in the majority, but not necessarily all, cases.

  90% of the orders are shipped via truck.

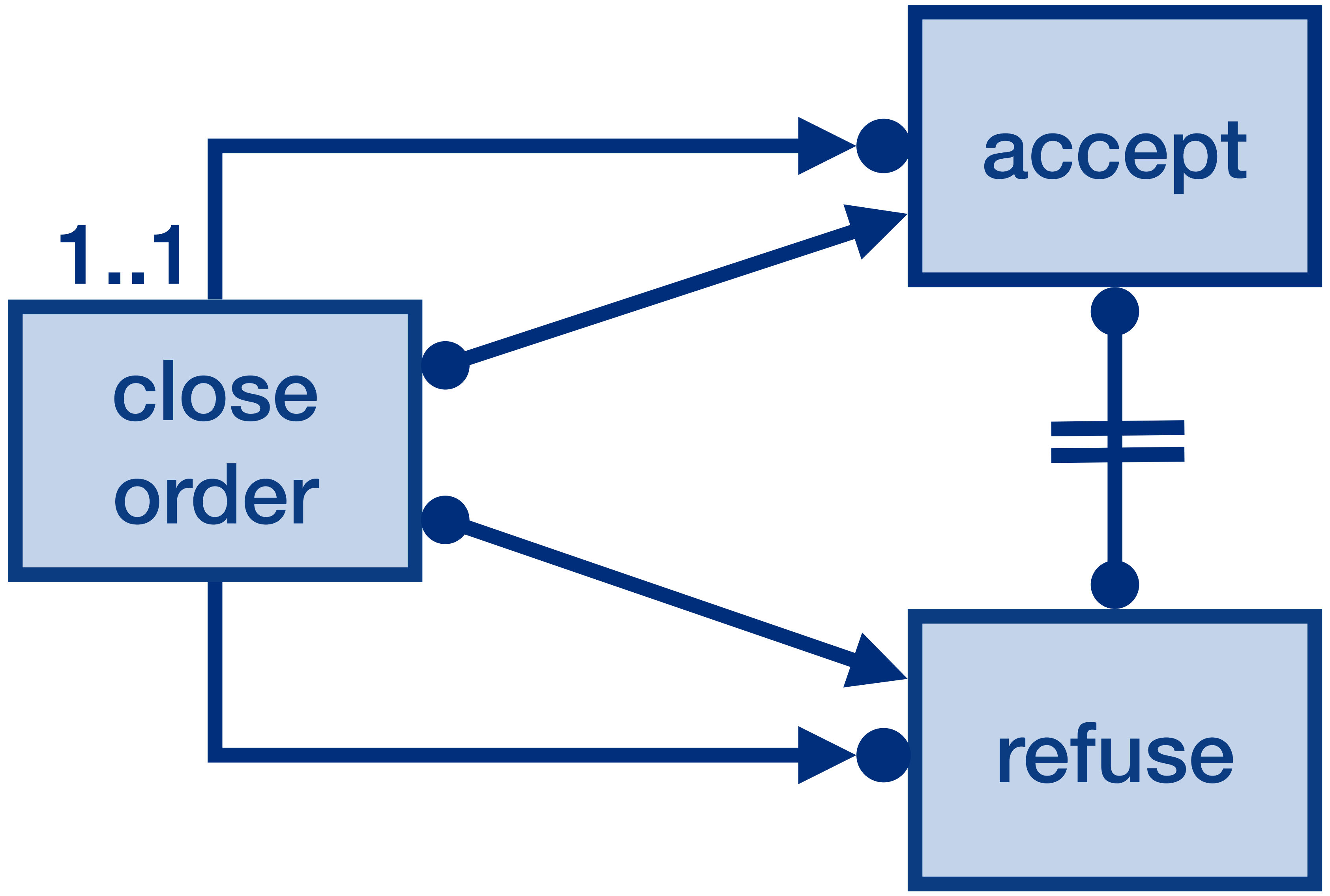
- **Outlier behaviors:** constraints that only apply to very few, but still conforming, cases.

  Only 1% of the orders are canceled after being paid.

- **Constraints involving external parties**: contain uncontrollable activities for which only partial guarantees can be given.

  In 8 cases out of 10, the customer accepts the order and also pays for it.

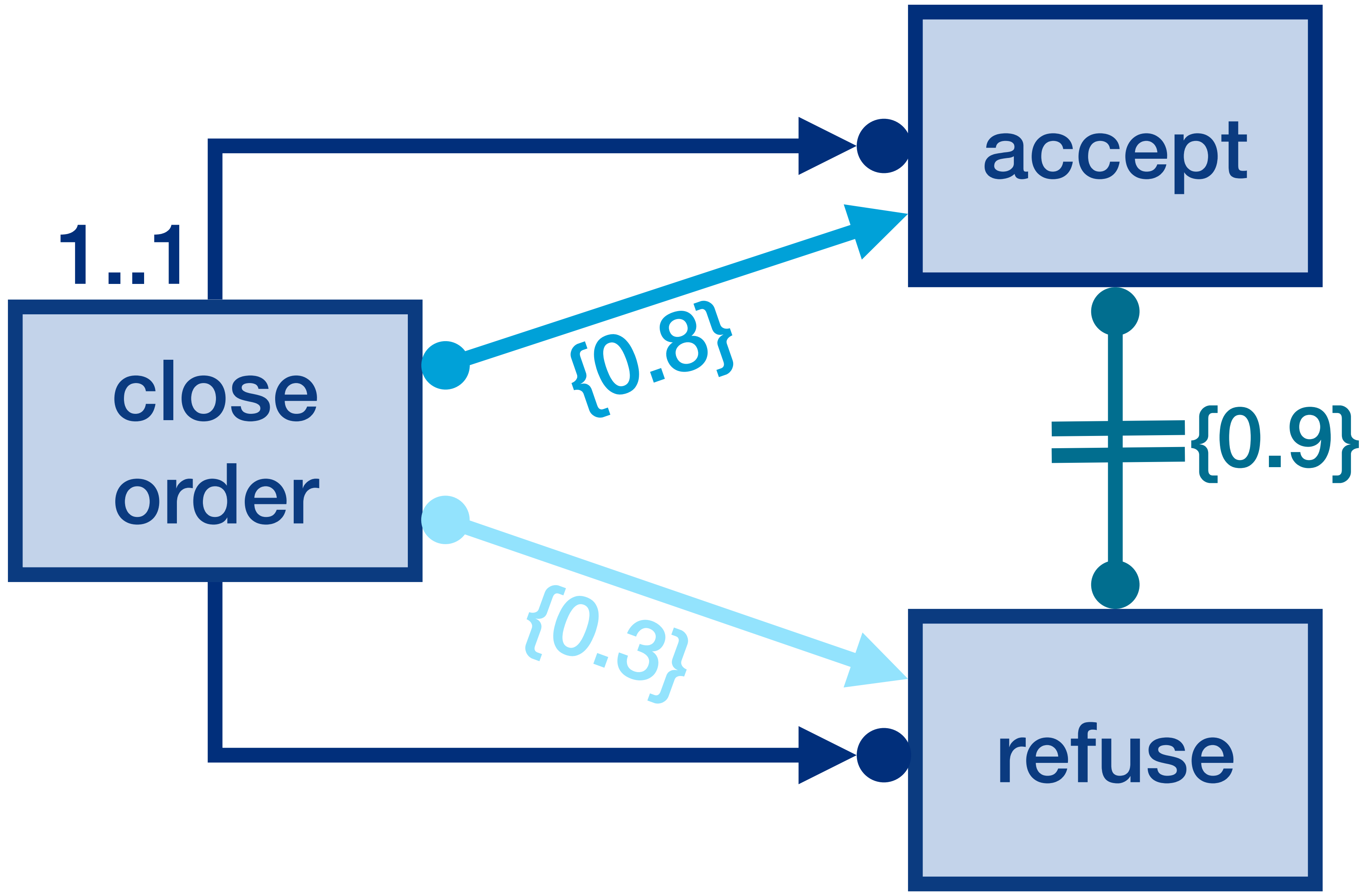
Declare is crisp



Crisp semantics: an execution trace conforms to the model if it satisfies every constraint in the model

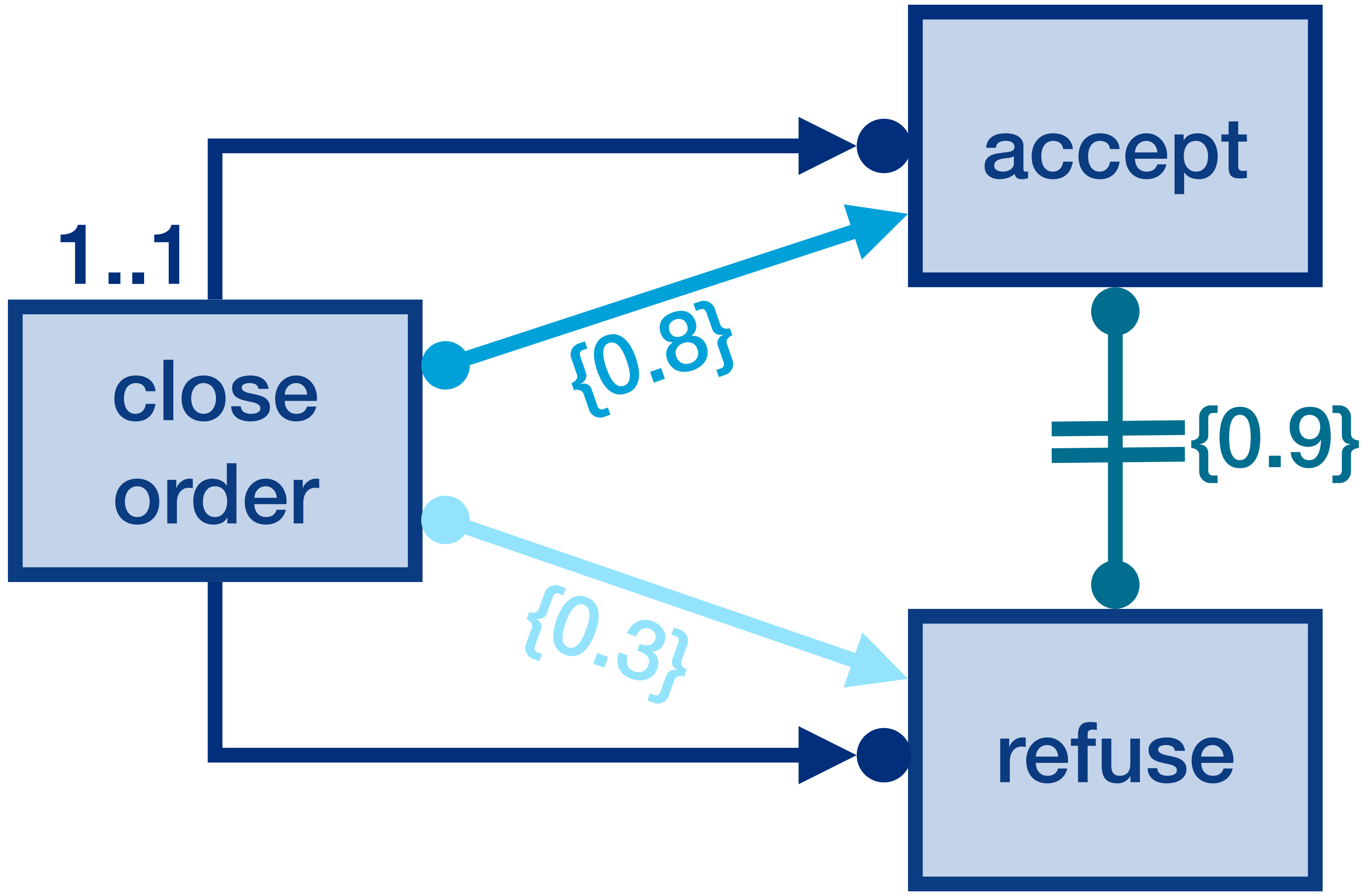
ProbDeclare

Crisp and uncertain constraints [____,BPM2020] [____,InfSys2022]



ProbDeclare

Crisp and uncertain constraints [____,BPM2020] [____,InfSys2022]



ProbDeclare constraint over Σ:

triple $\langle \varphi, \bowtie, p \rangle$

process condition: LTLf formula over Σ

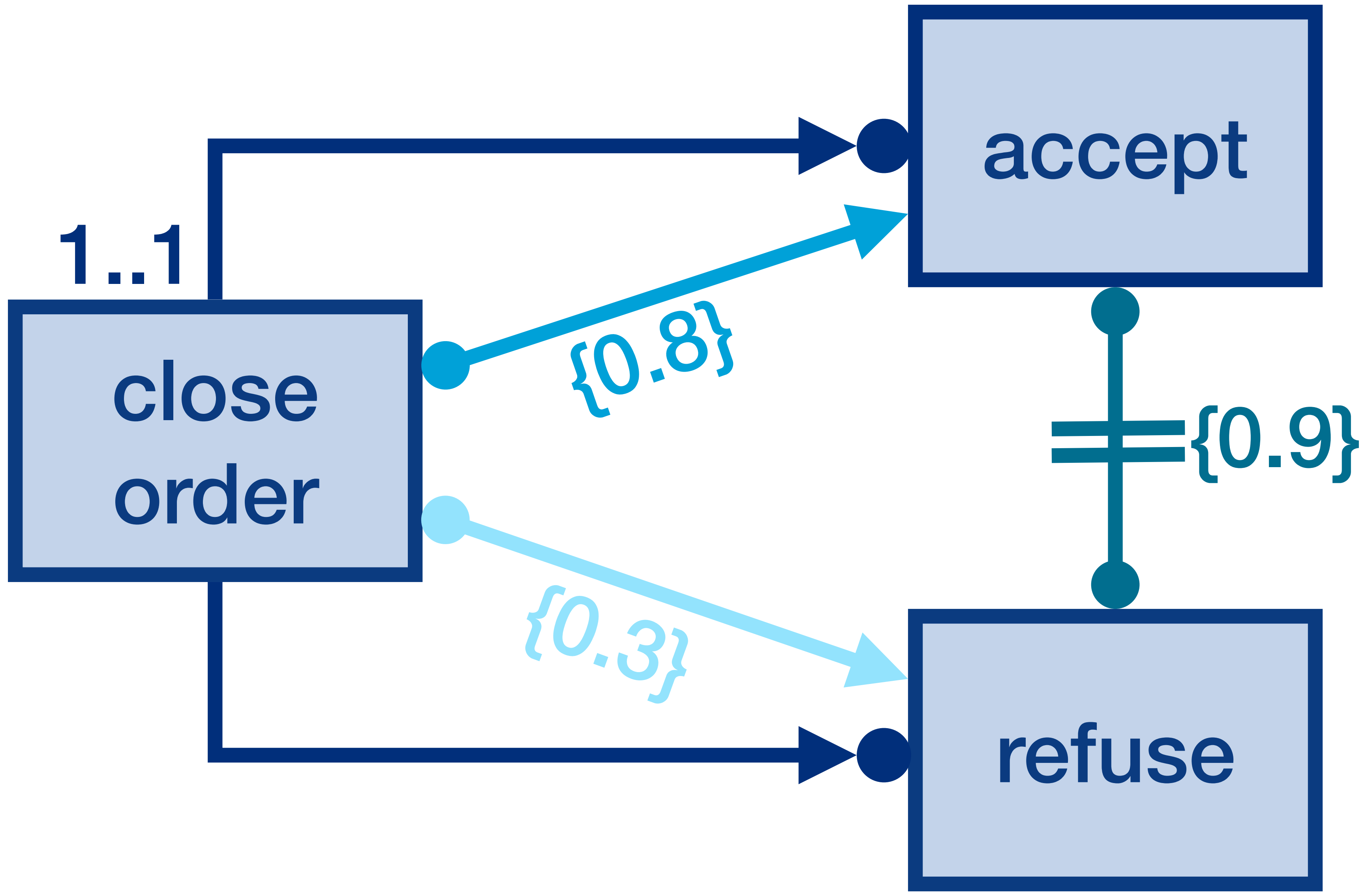
probability operator: $\{ =, \neq, \leq, \geq, <, > \}$

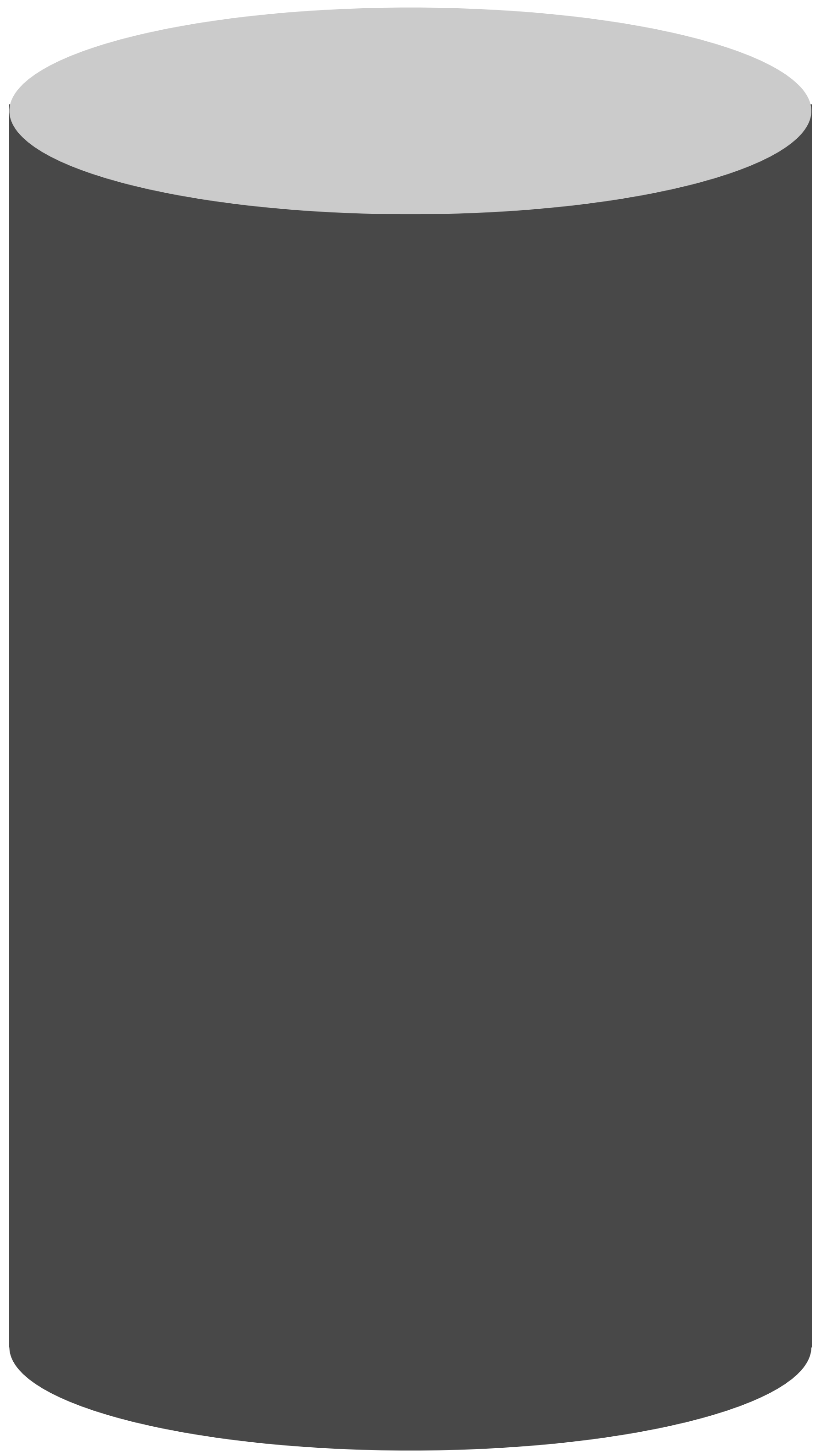
probability reference value: number in [0,1]

Well-behaved fragment of full probabilistic LTLf [____,AAAI2020]

ProbDeclare

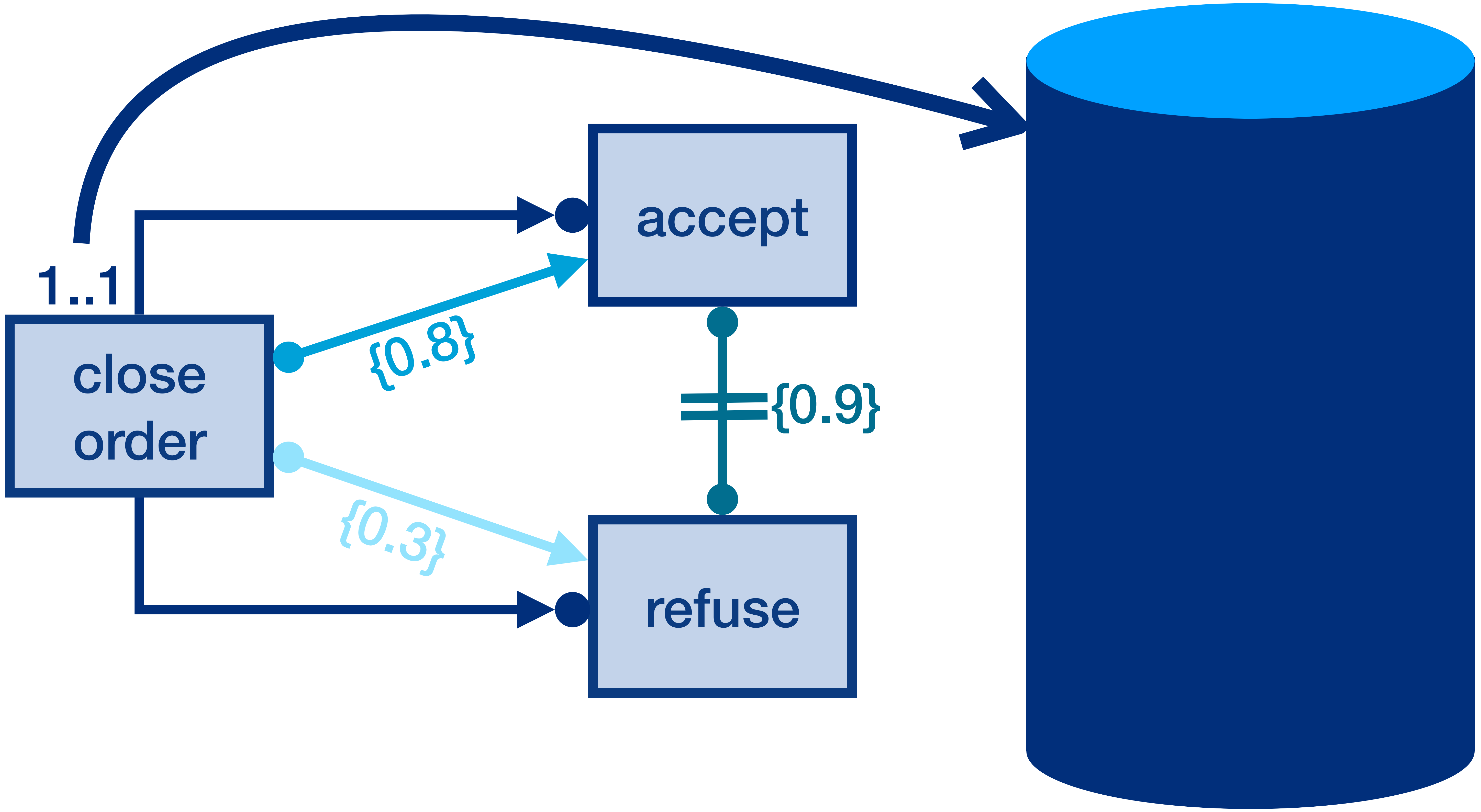
Crisp and uncertain constraints [____,BPM2020] [____,InfSys2022]





ProbDeclare

Crisp and uncertain constraints [____,BPM2020] [____,InfSys2022]

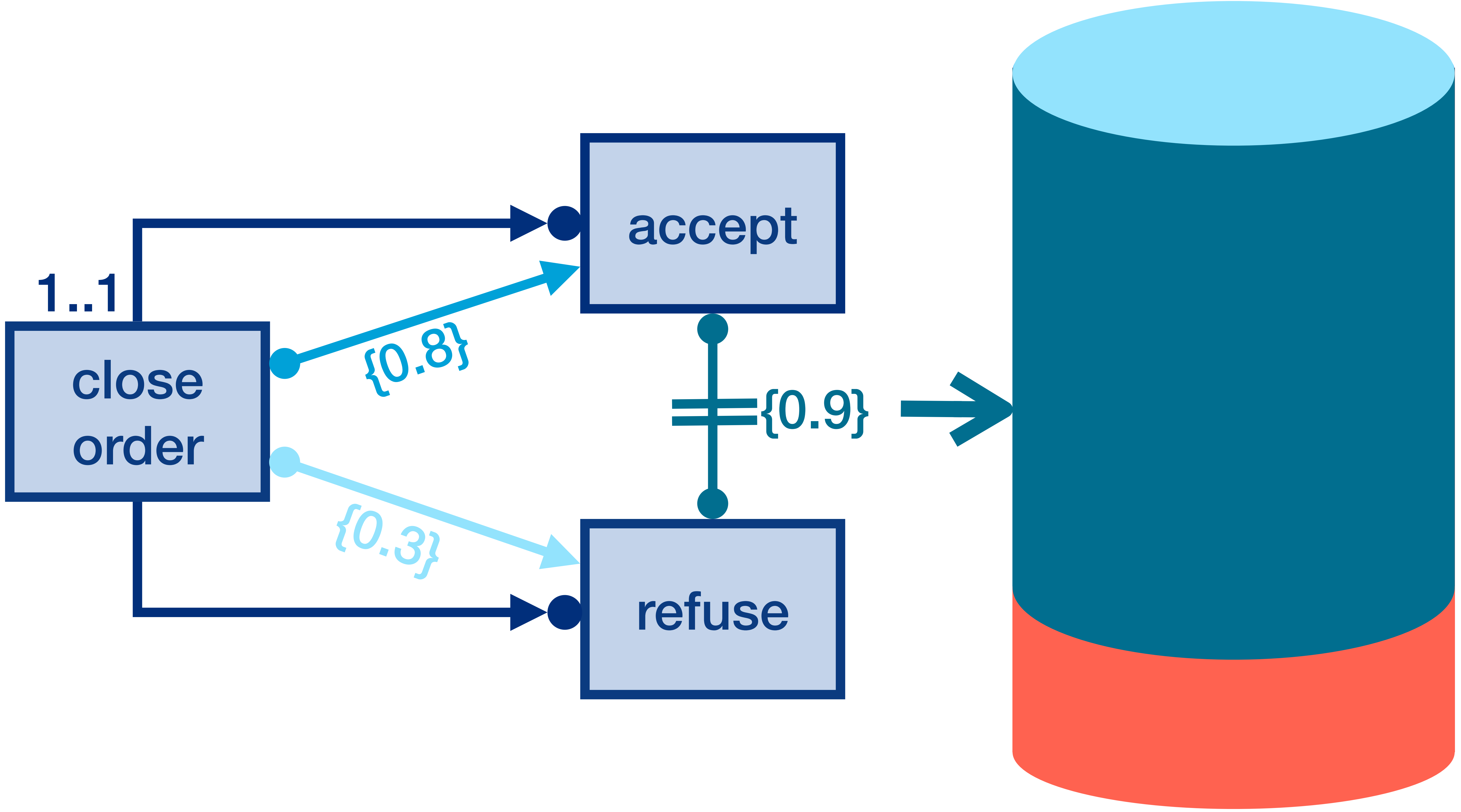


Crisp!

Each trace in the log contains exactly one **close order**

ProbDeclare

Crisp and uncertain constraints [____,BPM2020] [____,InfSys2022]

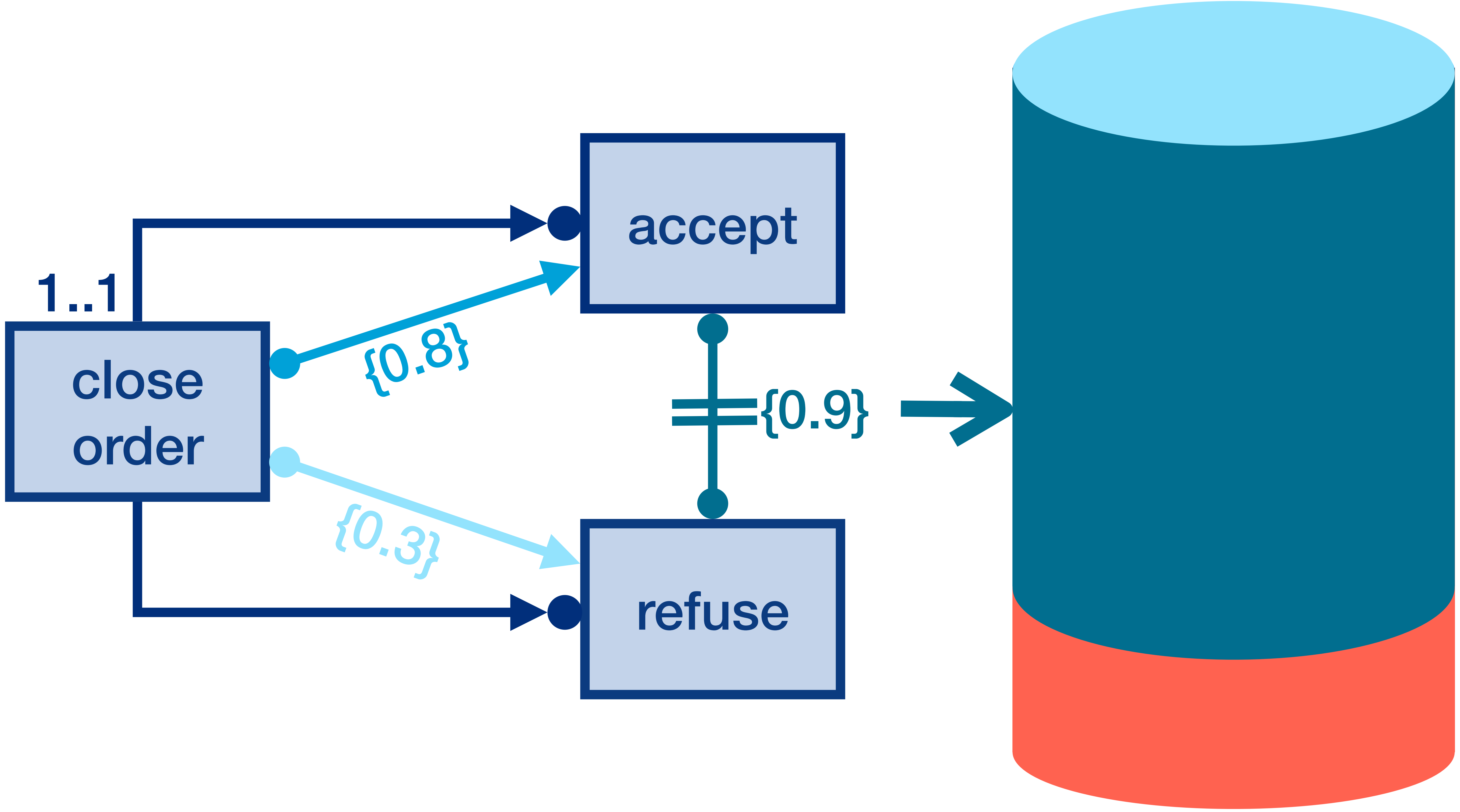


Uncertain!

90% traces are so that an order is not **accept**ed and **refuse**d.

ProbDeclare

Crisp and uncertain constraints [____,BPM2020] [____,InfSys2022]



Uncertain!

90% traces are so that an order is not **accept**ed and **refuse**d.

In 10% traces the seller **changes their mind**

From traces to stochastic languages and logs

A **stochastic language** over Σ is a function $\rho : \Sigma^* \rightarrow [0,1]$ such that $\sum_{\tau \in \Sigma^*} \rho(\tau) = 1$

- **finite** if finitely many traces get a non-zero probability

A log can be seen as a finite stochastic language (probabilities from frequencies)

Semantics of ProbDeclare

Stochastic language ρ **satisfies** ProbDeclare model if:

- for every **crisp constraint** φ and every trace $\tau \in \Sigma^*$ with non-zero probability, we have that $\tau \models \varphi$
- for every **probabilistic constraint** $\langle \varphi, \bowtie, p \rangle$, we have

$$\sum_{\tau \in \Sigma^*, \tau \models \varphi} \rho(\tau) \bowtie p$$

Semantics of ProbDeclare

Stochastic language ρ **satisfies** ProbDeclare model if:

- for every **crisp constraint** φ and every trace $\tau \in \Sigma^*$ with non-zero probability, we have that $\tau \models \varphi$
- for every **probabilistic constraint** $\langle \varphi, \bowtie, p \rangle$, we have

$$\sum_{\tau \in \Sigma^*, \tau \models \varphi} \rho(\tau) \bowtie p$$

Key challenge: again, interplay of constraints

Dealing with “n” probabilistic constraints

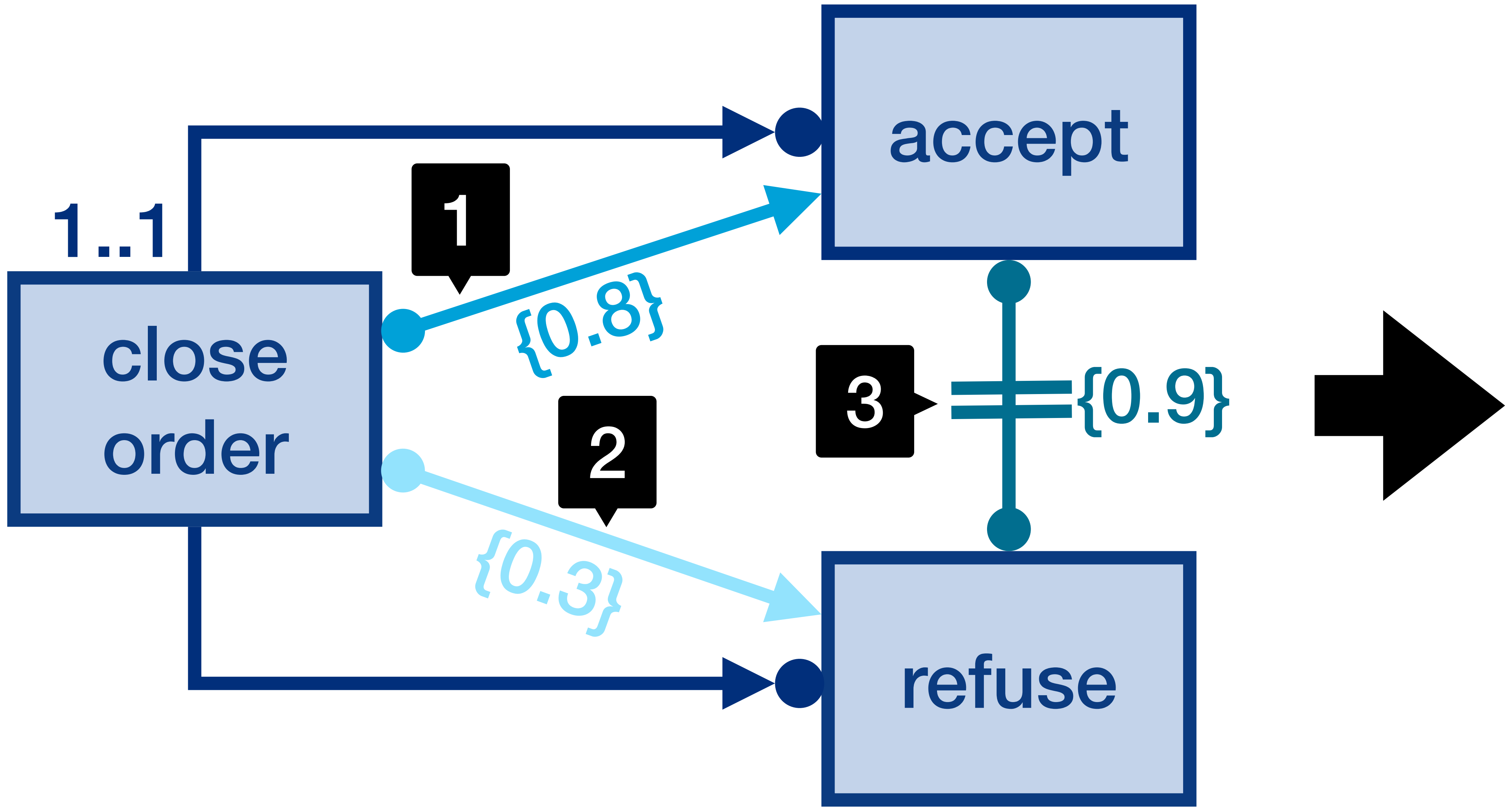
Constraint scenario

Declares which probabilistic constraints must hold, and which are violated

- Constraint violated <-> its negated version holds

Denotes a “process variant”

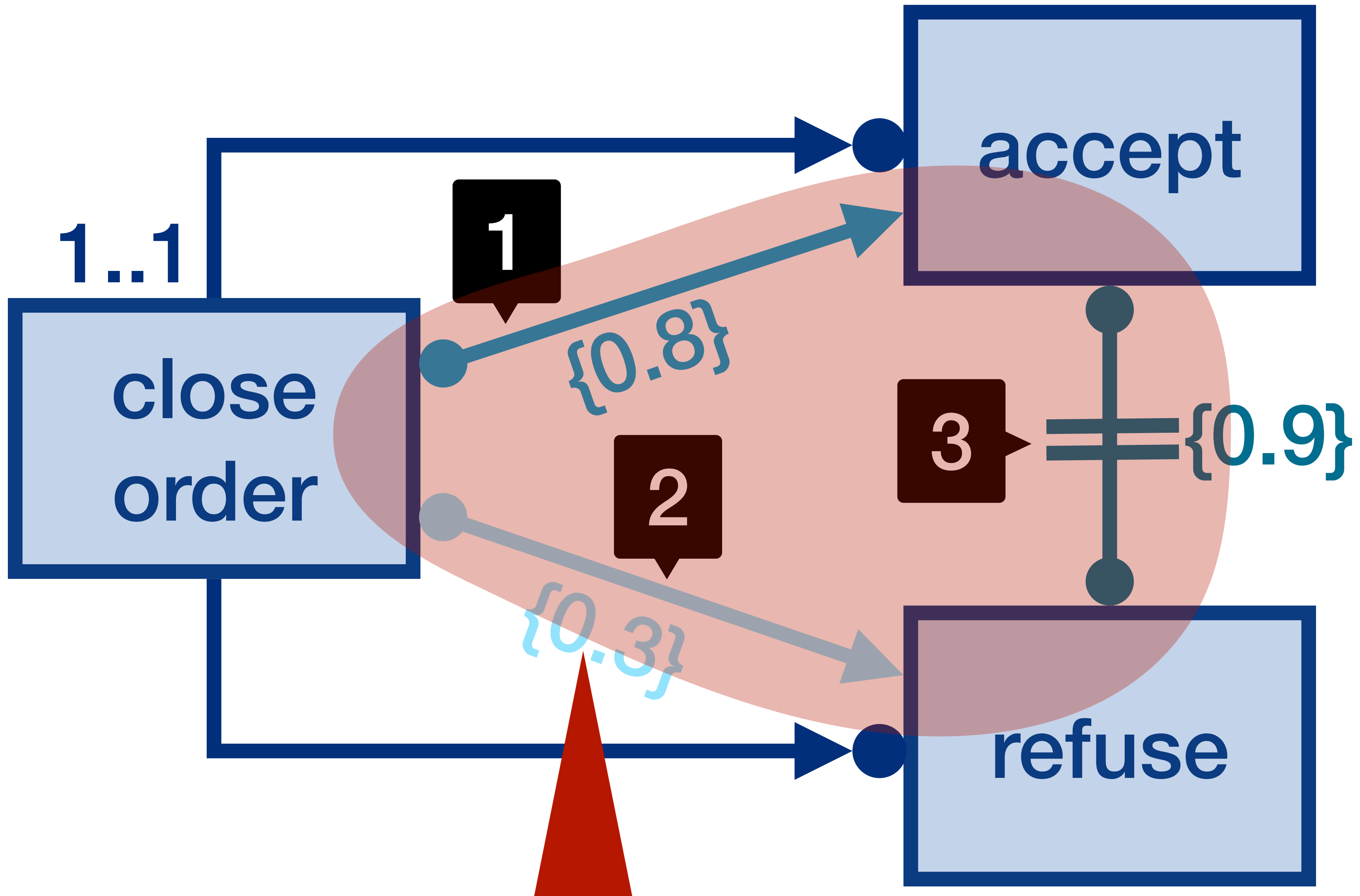
- All in all: up to 2^n scenarios, denoting different variants



8 scenarios		
(1)	(2)	(3)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Reasoning over scenarios is tricky

Interplay between logic and probabilities

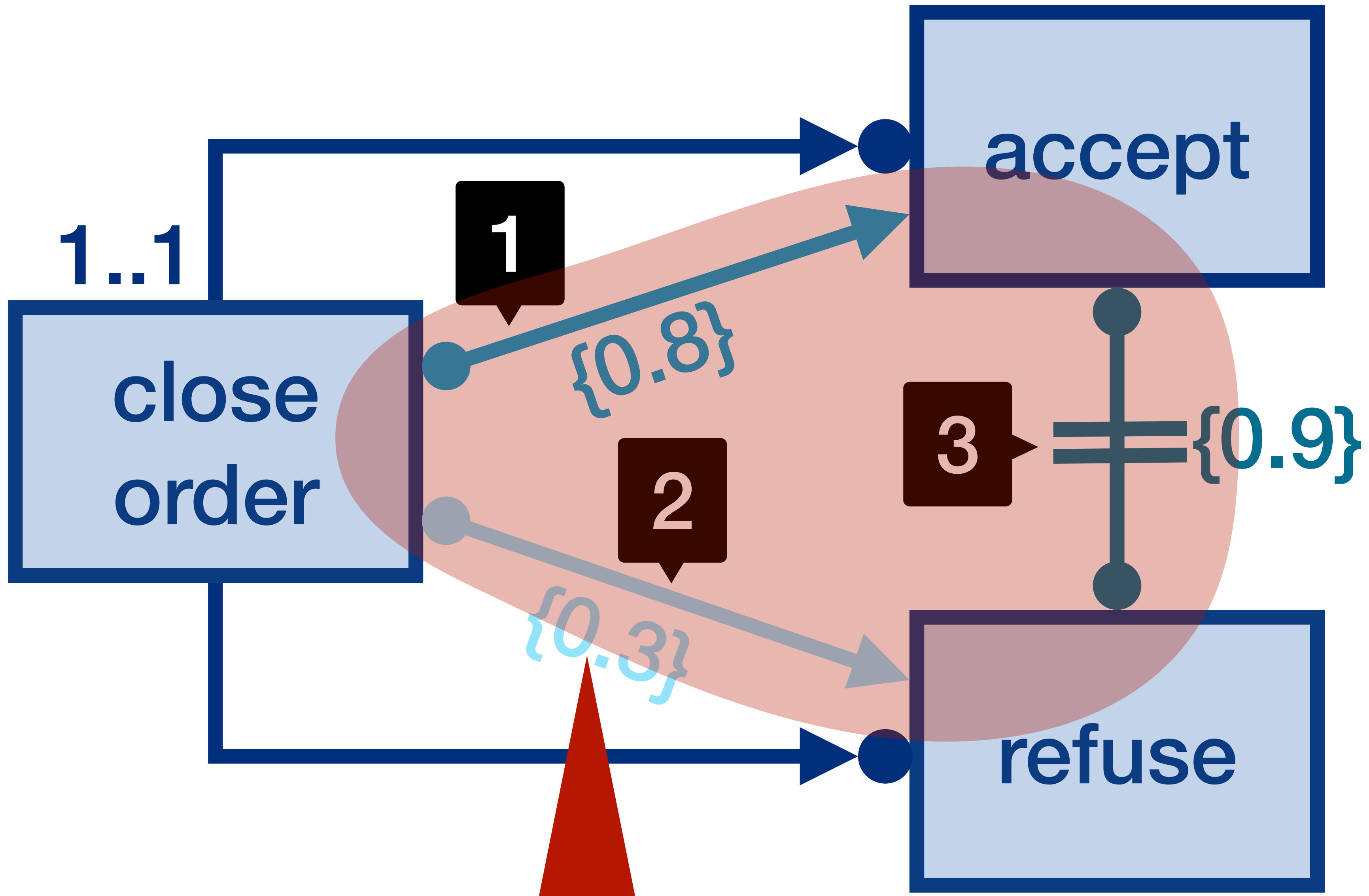


There cannot be traces that satisfy all constraints at once

8 scenarios		
(1)	(2)	(3)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Reasoning over scenarios is tricky

Interplay between logic and probabilities



There cannot be traces that satisfy all constraints at once

8 scenarios		
(1)	(2)	(3)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

inconsistent
-> no satisfying trace
-> 0 probability!

Logical reasoning within scenarios

LTLf and automata to the rescue

A scenario maps to an **LTLf characteristic formula**

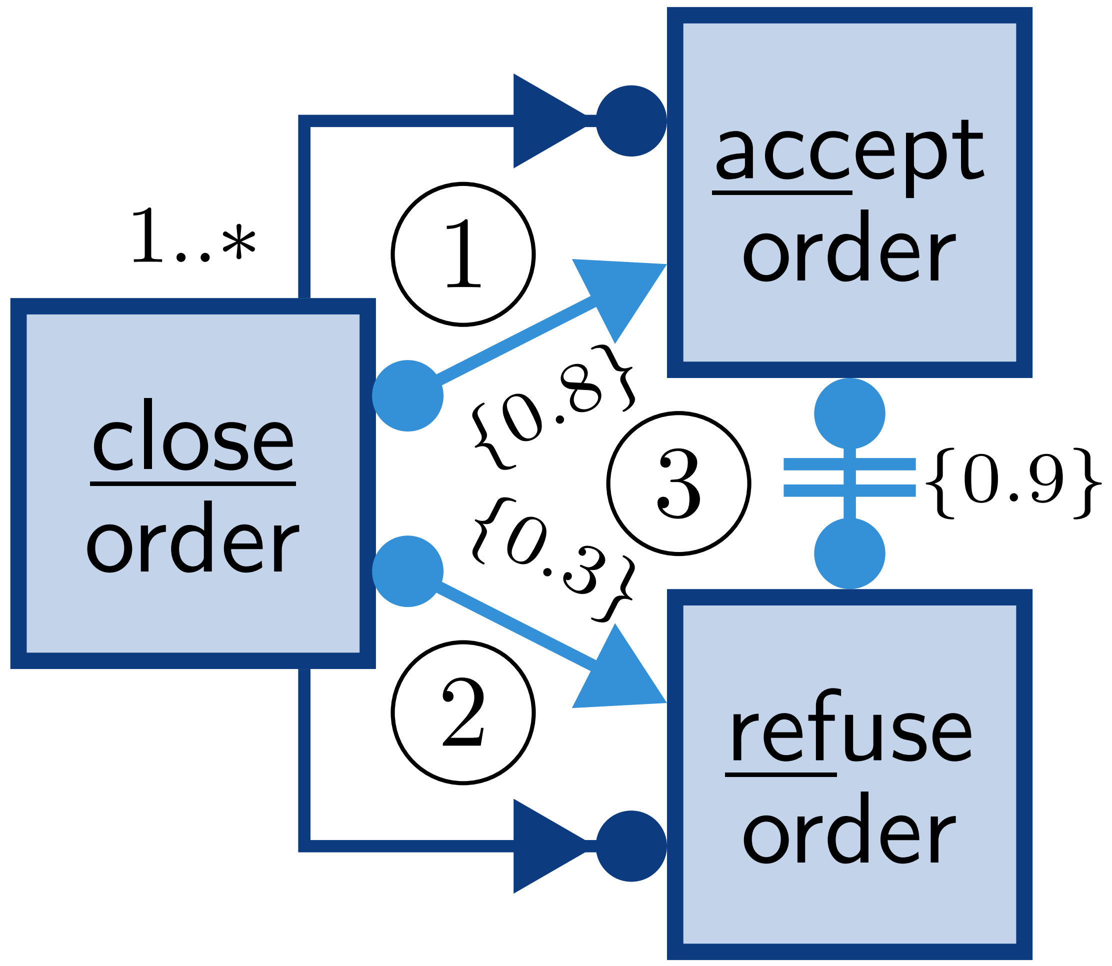
- Conjunction of formulae, one per constraint…
- Does the constraint hold in the scenario?
  - **Y** -> take its **LTLf process condition**
  - **N** -> take its **negation**

$$\Phi(S^M_{b_1 \cdots b_n}) = \bigwedge_{\psi \in \mathcal{C}} \psi \wedge \bigwedge_{i \in \{1,\ldots,n\}} \begin{cases} \varphi_i & \text{if } b_i = 1 \\ \neg\varphi_i & \text{if } b_i = 0 \end{cases}$$

Reasoning via automata, as for standard LTLf

In our example…

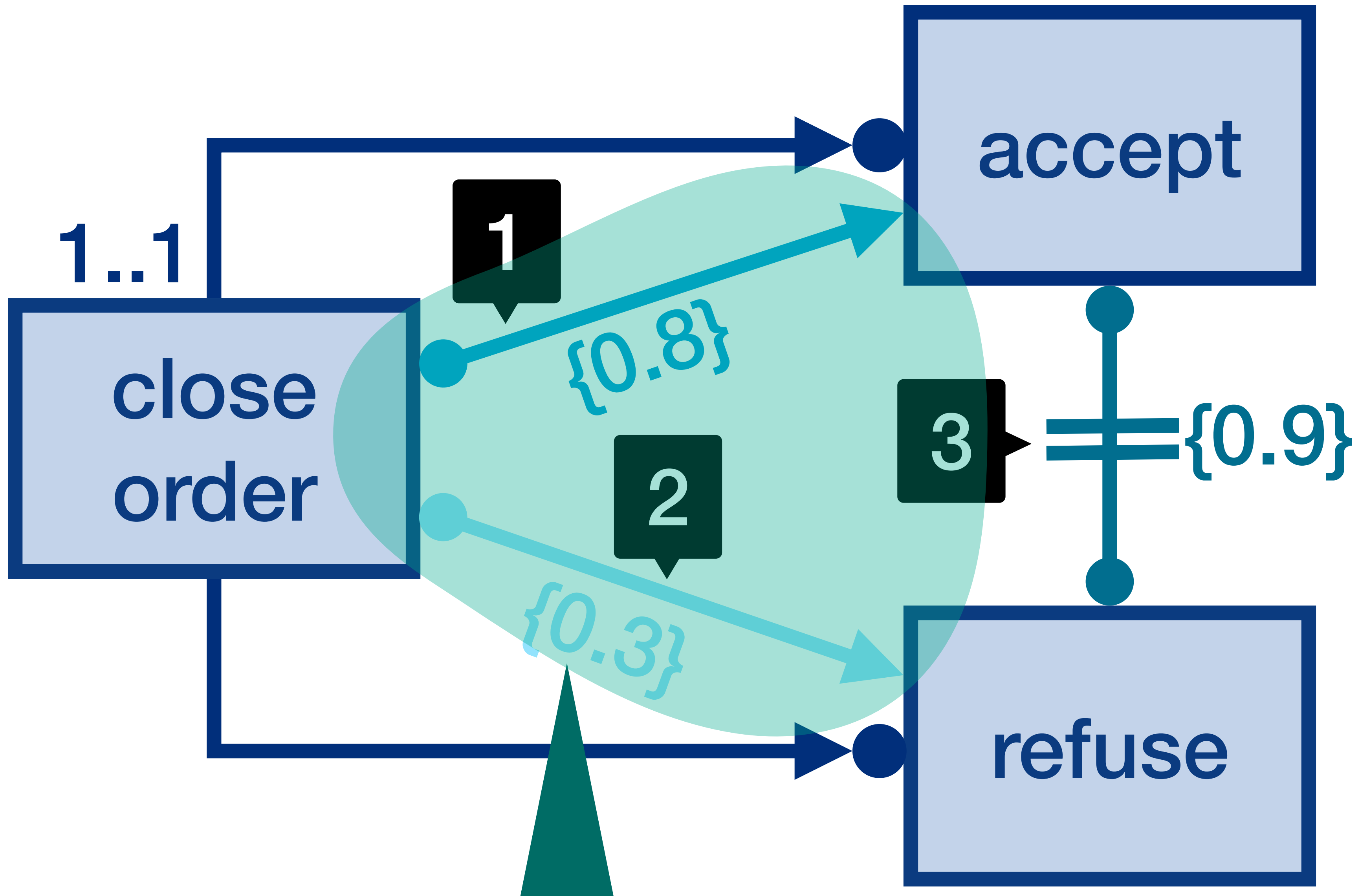
Which scenarios are consistent?



	①	②	③	CONSISTENT?
S_{000}	$\Diamond(\text{close} \wedge \neg\bigcirc\Diamond\text{acc})$	$\Diamond(\text{close} \wedge \neg\bigcirc\Diamond\text{ref})$	$\Diamond\text{acc} \wedge \Diamond\text{refuse}$	no
S_{001}	$\Diamond(\text{close} \wedge \neg\bigcirc\Diamond\text{acc})$	$\Diamond(\text{close} \wedge \neg\bigcirc\Diamond\text{ref})$	$\neg(\Diamond\text{acc} \wedge \Diamond\text{refuse})$	**yes**
S_{010}	$\Diamond(\text{close} \wedge \neg\bigcirc\Diamond\text{acc})$	$\Box(\text{close} \rightarrow \bigcirc\Diamond\text{ref})$	$\Diamond\text{acc} \wedge \Diamond\text{refuse}$	no
S_{011}	$\Diamond(\text{close} \wedge \neg\bigcirc\Diamond\text{acc})$	$\Box(\text{close} \rightarrow \bigcirc\Diamond\text{ref})$	$\neg(\Diamond\text{acc} \wedge \Diamond\text{refuse})$	**yes**
S_{100}	$\Box(\text{close} \rightarrow \bigcirc\Diamond\text{acc})$	$\Diamond(\text{close} \wedge \neg\bigcirc\Diamond\text{ref})$	$\Diamond\text{acc} \wedge \Diamond\text{refuse}$	no
S_{101}	$\Box(\text{close} \rightarrow \bigcirc\Diamond\text{acc})$	$\Diamond(\text{close} \wedge \neg\bigcirc\Diamond\text{ref})$	$\neg(\Diamond\text{acc} \wedge \Diamond\text{refuse})$	**yes**
S_{110}	$\Box(\text{close} \rightarrow \bigcirc\Diamond\text{acc})$	$\Box(\text{close} \rightarrow \bigcirc\Diamond\text{ref})$	$\Diamond\text{acc} \wedge \Diamond\text{refuse}$	**yes**
S_{111}	$\Box(\text{close} \rightarrow \bigcirc\Diamond\text{acc})$	$\Box(\text{close} \rightarrow \bigcirc\Diamond\text{ref})$	$\neg(\Diamond\text{acc} \wedge \Diamond\text{refuse})$	no

Reasoning over scenarios is tricky

Interplay between logic and probabilities

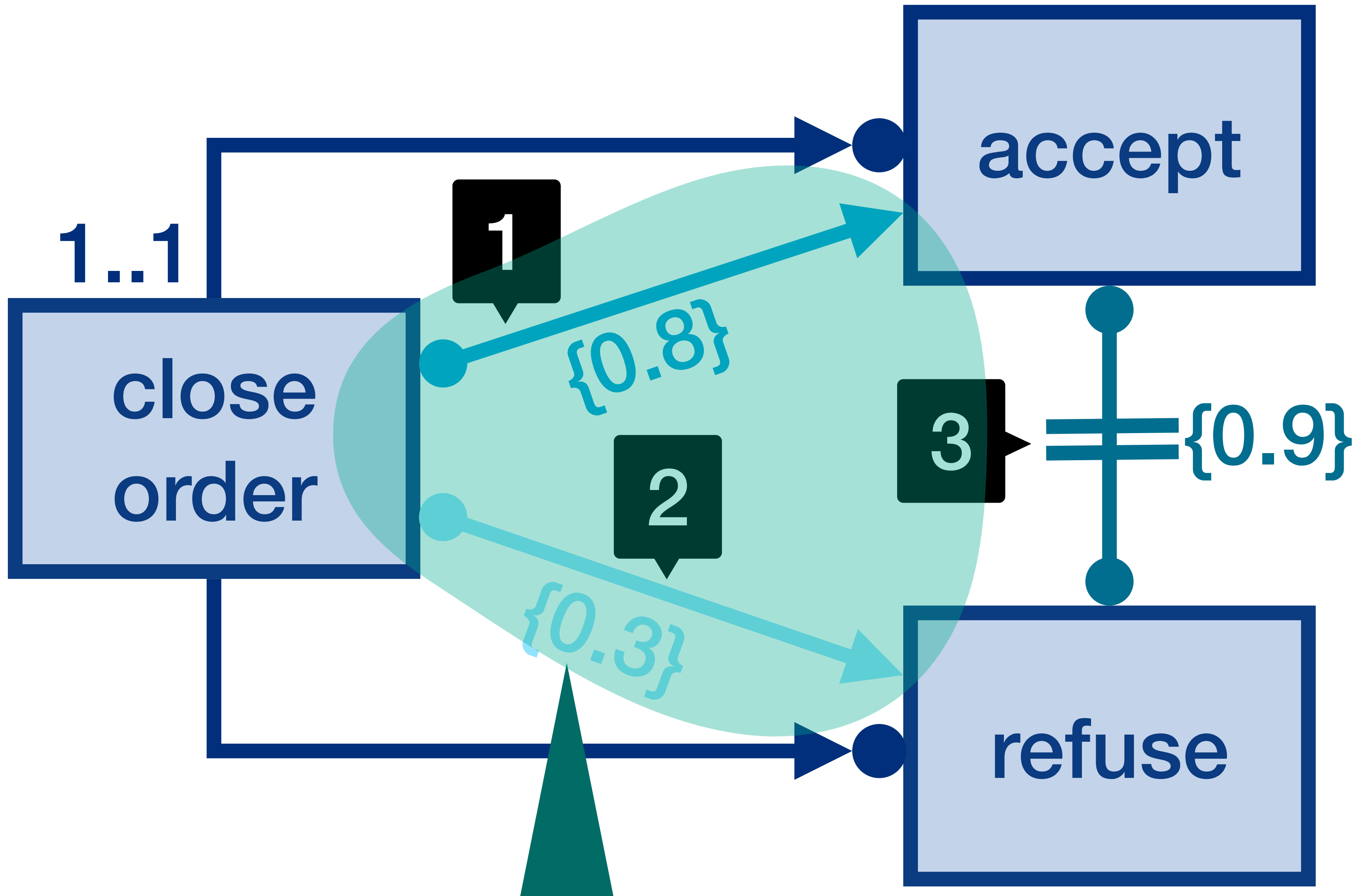


0.8+0.3 > 1
-> there must be traces where a closed order is accepted and refused.

8 scenarios		
(1)	(2)	(3)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Reasoning over scenarios is tricky

Interplay between logic and probabilities



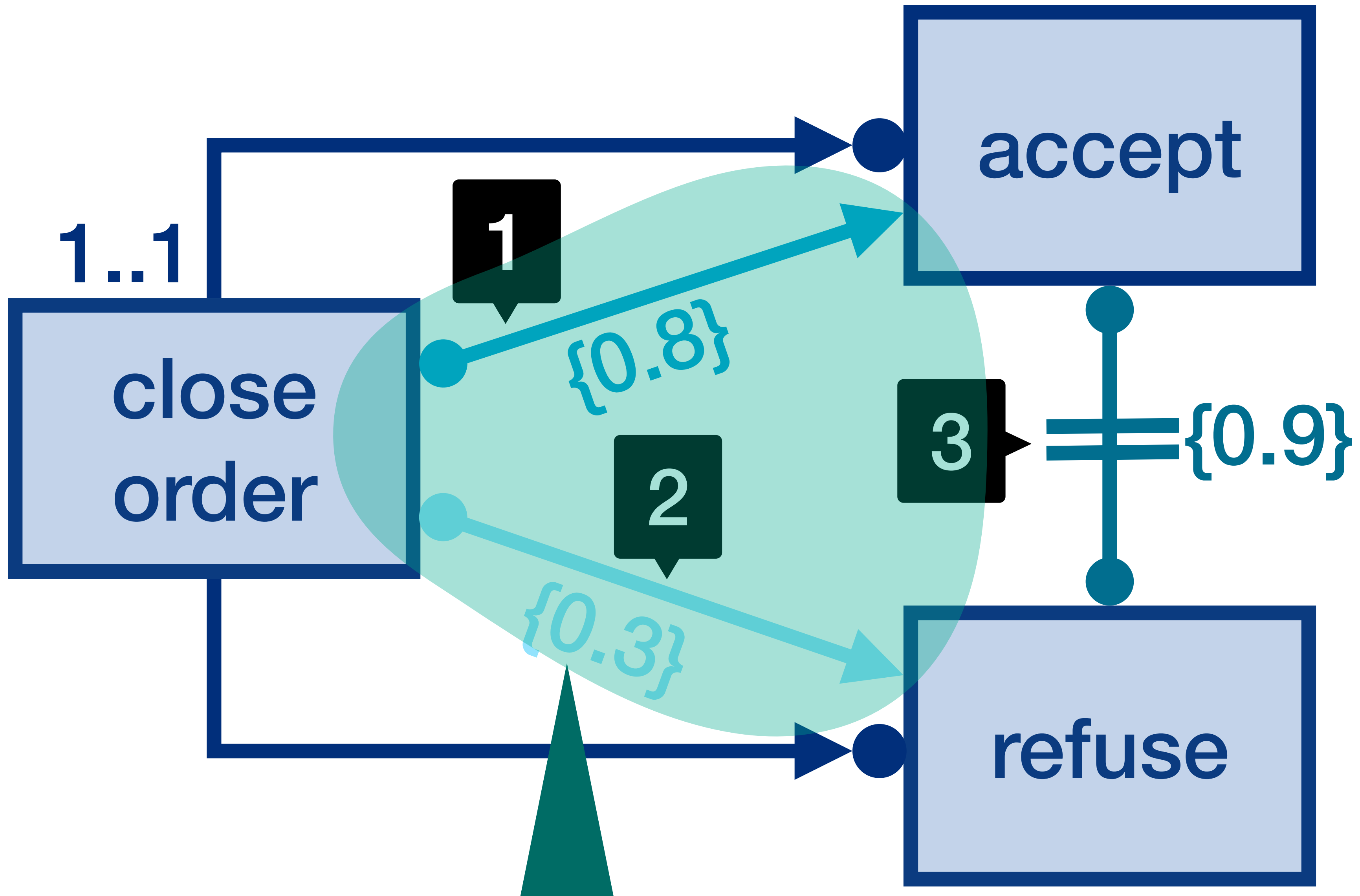
there must be traces where accept and refuse coexist

0.8+0.3 > 1
-> there must be traces where a closed order is accepted and refused.

8 scenarios		
(1)	(2)	(3)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Reasoning over scenarios is tricky

Interplay between logic and probabilities



there must be traces where accept and refuse coexist

8 scenarios		
(1)	(2)	(3)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

0.8+0.3 > 1
-> there must be traces where a closed order is accepted and refused.

Should have a non-zero probability (if constraint values agree)

The true meaning of a ProbDeclare model

From probabilistic constraints to scenario probability distributions

With n scenarios: x_i with $i \in \{0,\ldots,2^{n-1}\}$ denotes the probability that a trace belongs to scenario i

ProbDeclare model: constrains the legal probability distributions over scenarios

$$x_i \geq 0 \qquad 0 \leq i < 2^n$$

$$\Big(\sum_{i=0}^{2^n-1} x_i\Big) = 1$$

$$\Big(\sum_{\substack{i \in \{0,\ldots,2^n-1\},\\ j\text{th position of } i \text{ is } 1}} x_i\Big) \bowtie_j p_j \qquad 0 \leq j < n$$

$$x_i = 0 \qquad 0 \leq i < 2^n, \text{scenario } S_i \text{ is inconsistent}$$

The true meaning of a ProbDeclare model

From probabilistic constraints to scenario probability distributions

With n scenarios: x_i with $i \in \{0,\ldots,2^{n-1}\}$ denotes the probability that a trace belongs to scenario i

ProbDeclare model: constrains the legal probability distributions over scenarios

$$x_i \geq 0 \qquad 0 \leq i < 2^n$$

$$\left(\sum_{i=0}^{2^n-1} x_i\right) = 1$$

$$\left(\sum_{\substack{i \in \{0,\ldots,2^n-1\},\\ j\text{th position of } i \text{ is } 1}} x_i\right) \bowtie_j p_j \qquad 0 \leq j < n$$

$$x_i = 0 \qquad 0 \leq i < 2^n, \text{scenario } S_i \text{ is inconsistent}$$

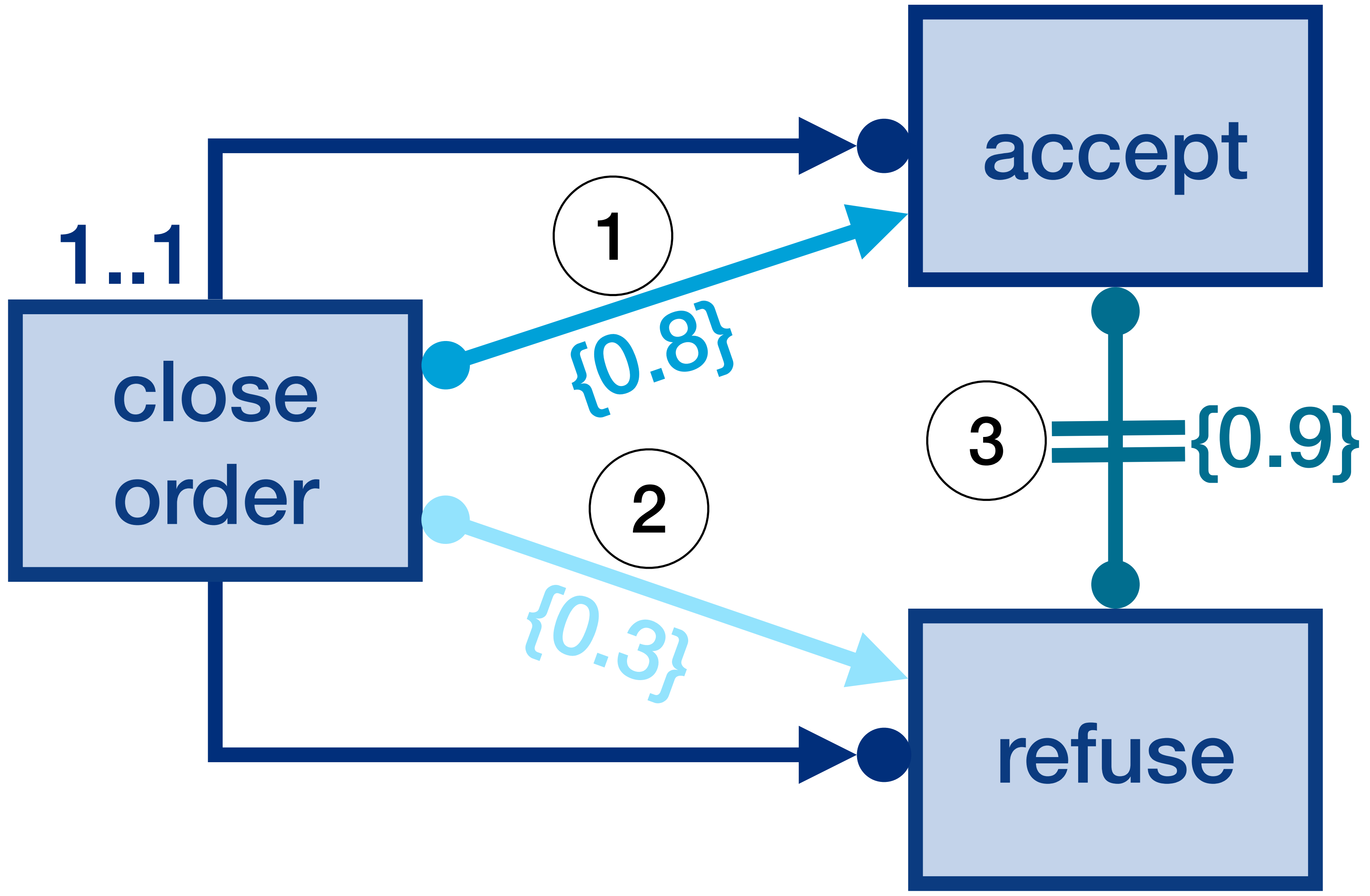
One solution
-> a fixed probability distribution

(Possibly infinitely) many solutions
-> family of probability distributions

No solution
-> inconsistent specification

Computing probability distributions

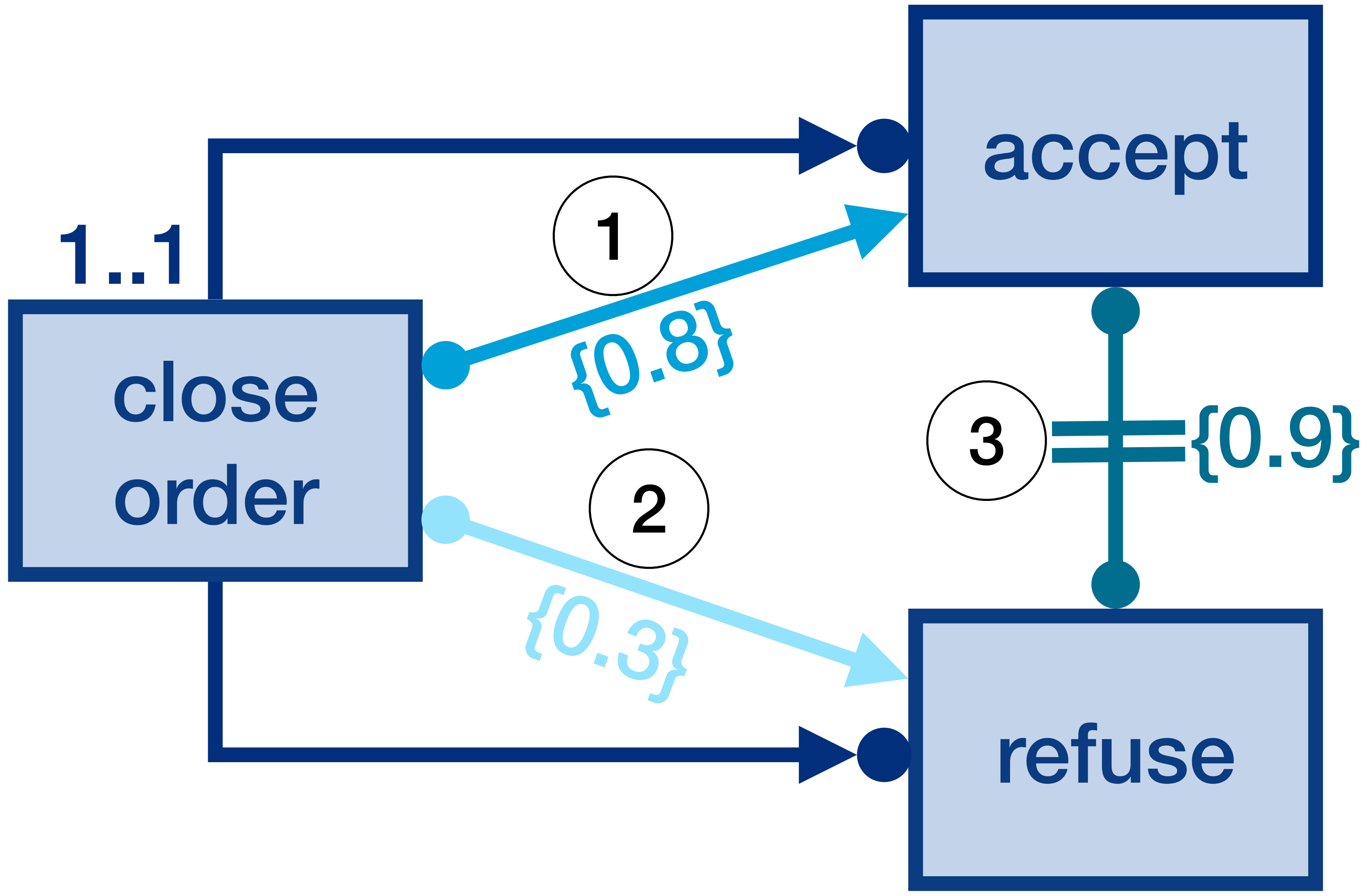
1. check for consistency



scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Computing probability distributions

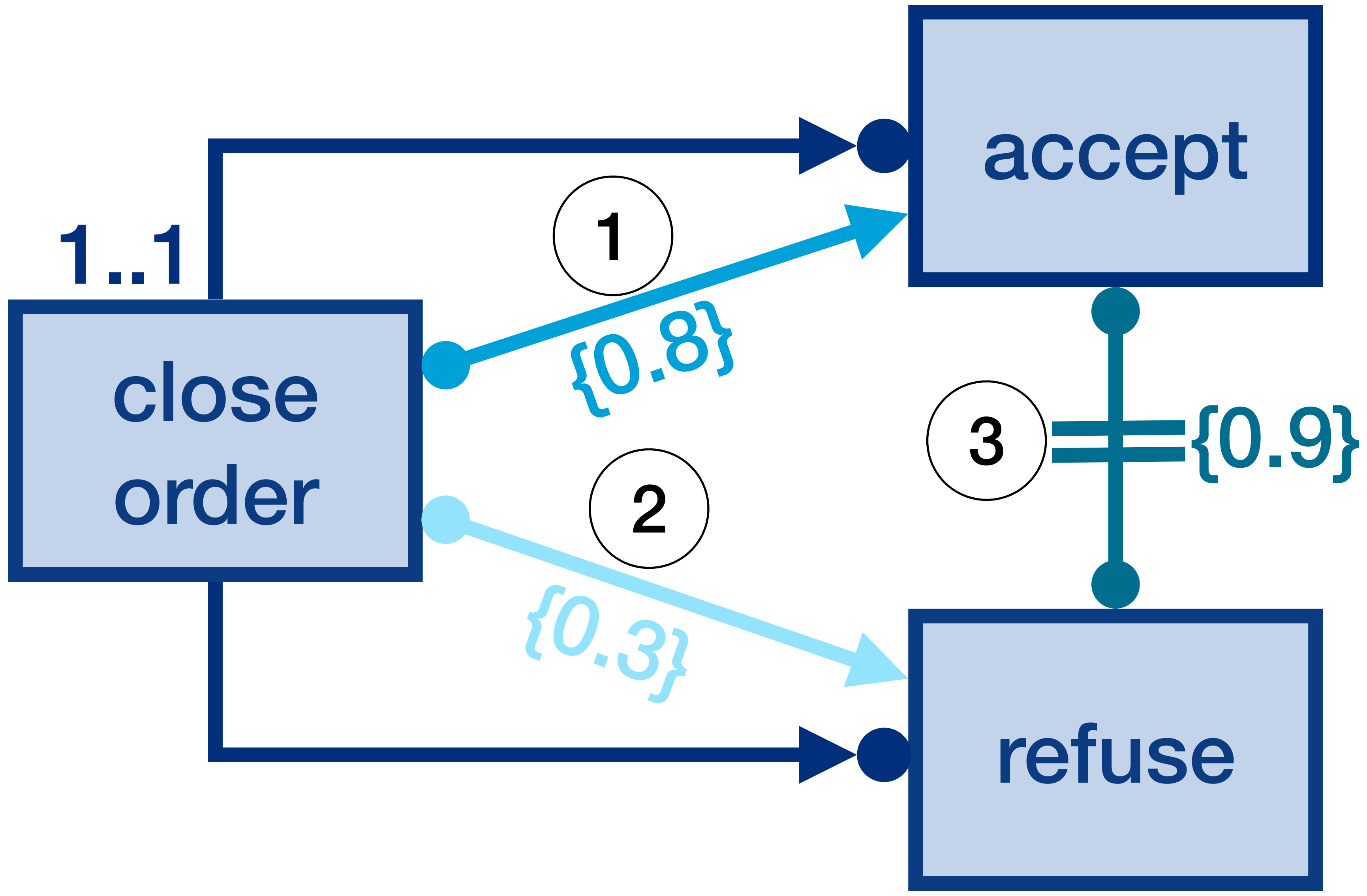
1. check for consistency



scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	
0	0	1	Y	
0	1	0	N	
0	1	1	Y	
1	0	0	N	
1	0	1	Y	
1	1	0	Y	
1	1	1	N	

Computing probability distributions

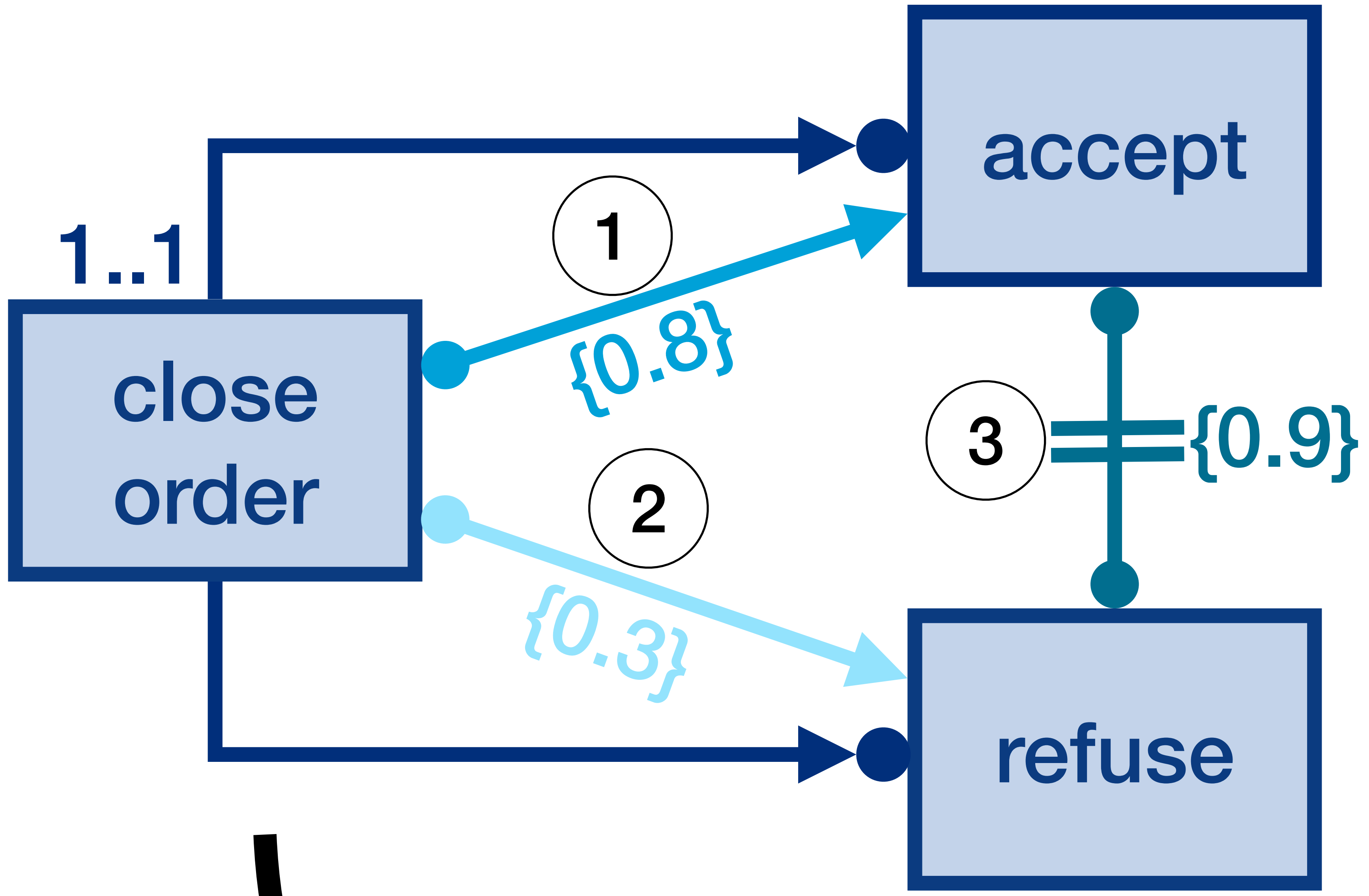
1. check for consistency



scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	
0	1	0	N	0
0	1	1	Y	
1	0	0	N	0
1	0	1	Y	
1	1	0	Y	
1	1	1	N	0

Computing probability distributions

2. set up system of inequalities

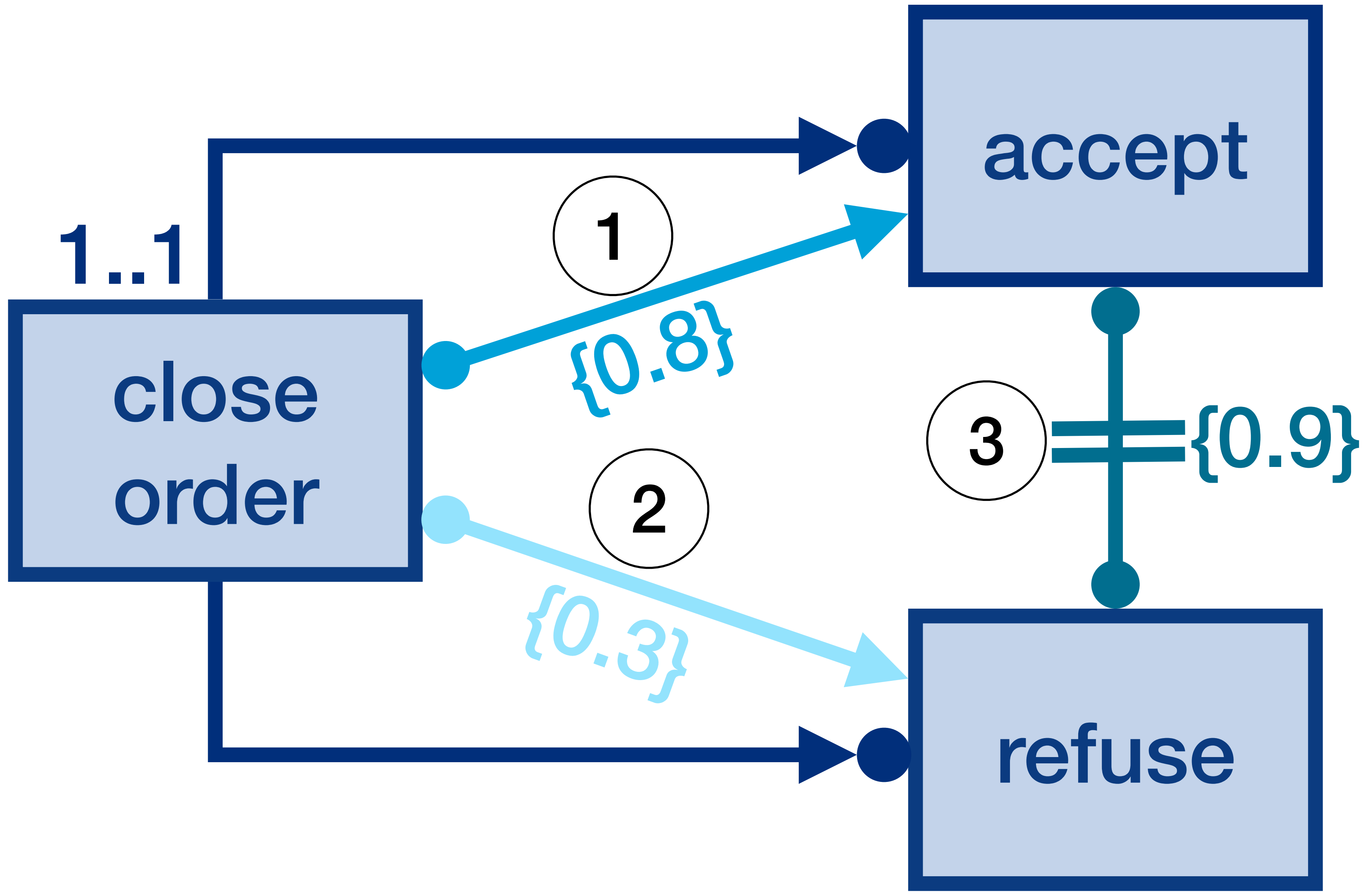


scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	
0	1	0	N	0
0	1	1	Y	
1	0	0	N	0
1	0	1	Y	
1	1	0	Y	
1	1	1	N	0

$$
\begin{aligned}
x_{001} + x_{011} + x_{101} + x_{110} &= 1 \\
x_{101} + x_{110} &= 0.8 \\
x_{011} + x_{110} &= 0.3 \\
x_{001} + x_{011} + x_{101} &= 0.9
\end{aligned}
$$

Computing probability distributions

3. solve

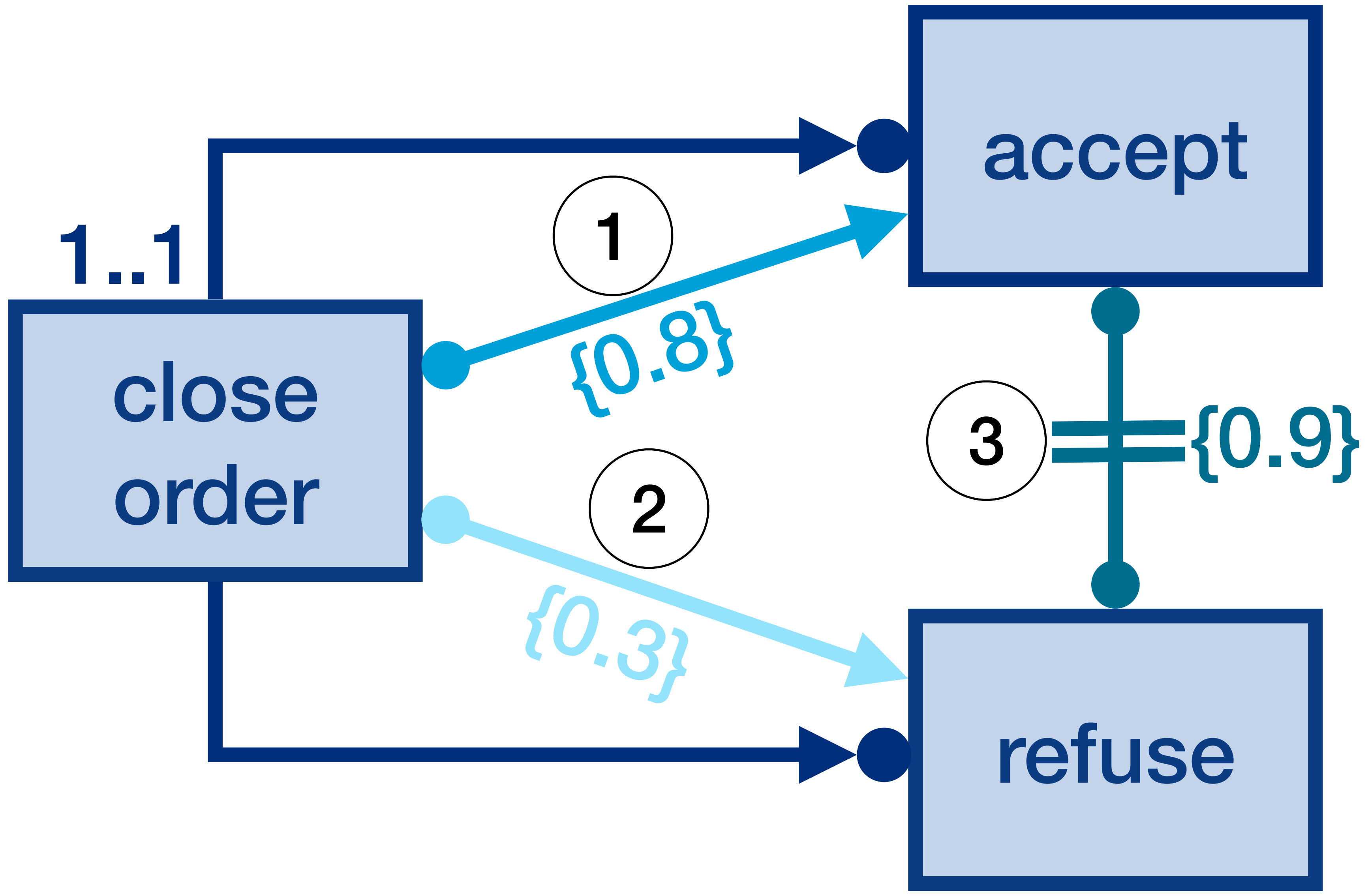


scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0

$$\begin{array}{ccccccccc}
x_{001} & + & x_{011} & + & x_{101} & + & x_{110} & = & 1 \\
 & & & & x_{101} & + & x_{110} & = & 0.8 \\
 & & x_{011} & & & + & x_{110} & = & 0.3 \\
x_{001} & + & x_{011} & + & x_{101} & & & = & 0.9
\end{array}$$

Computing probability distributions

3. solve

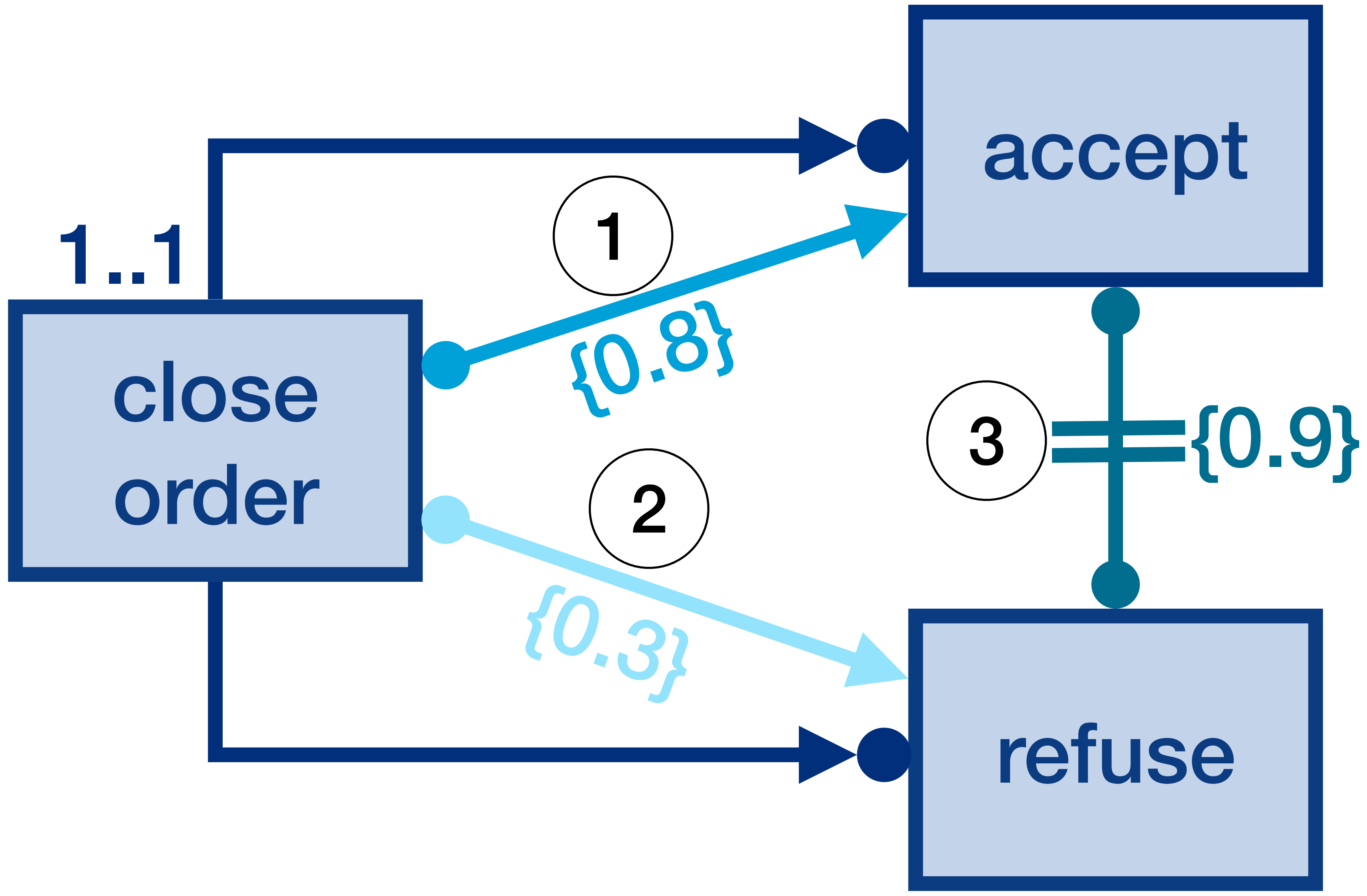


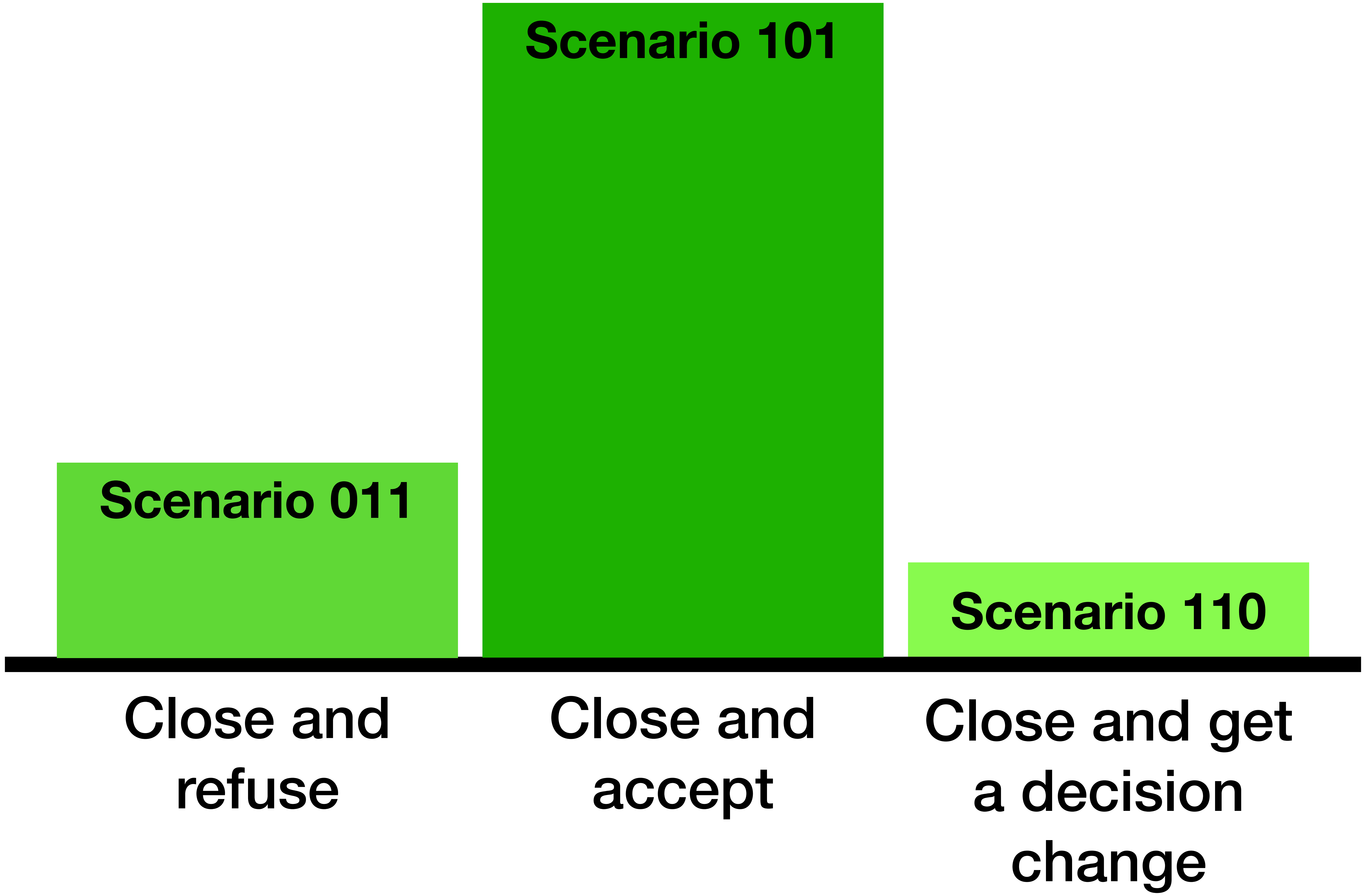
scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0

$$
\begin{array}{ccccccccccc}
x_{001} & + & x_{011} & + & x_{101} & + & x_{110} & = & 1 \\
 & & & & x_{101} & + & x_{110} & = & 0.8 \\
 & & x_{011} & & & + & x_{110} & = & 0.3 \\
x_{001} & + & x_{011} & + & x_{101} & & & = & 0.9
\end{array}
$$

Computing probability distributions

3. solve

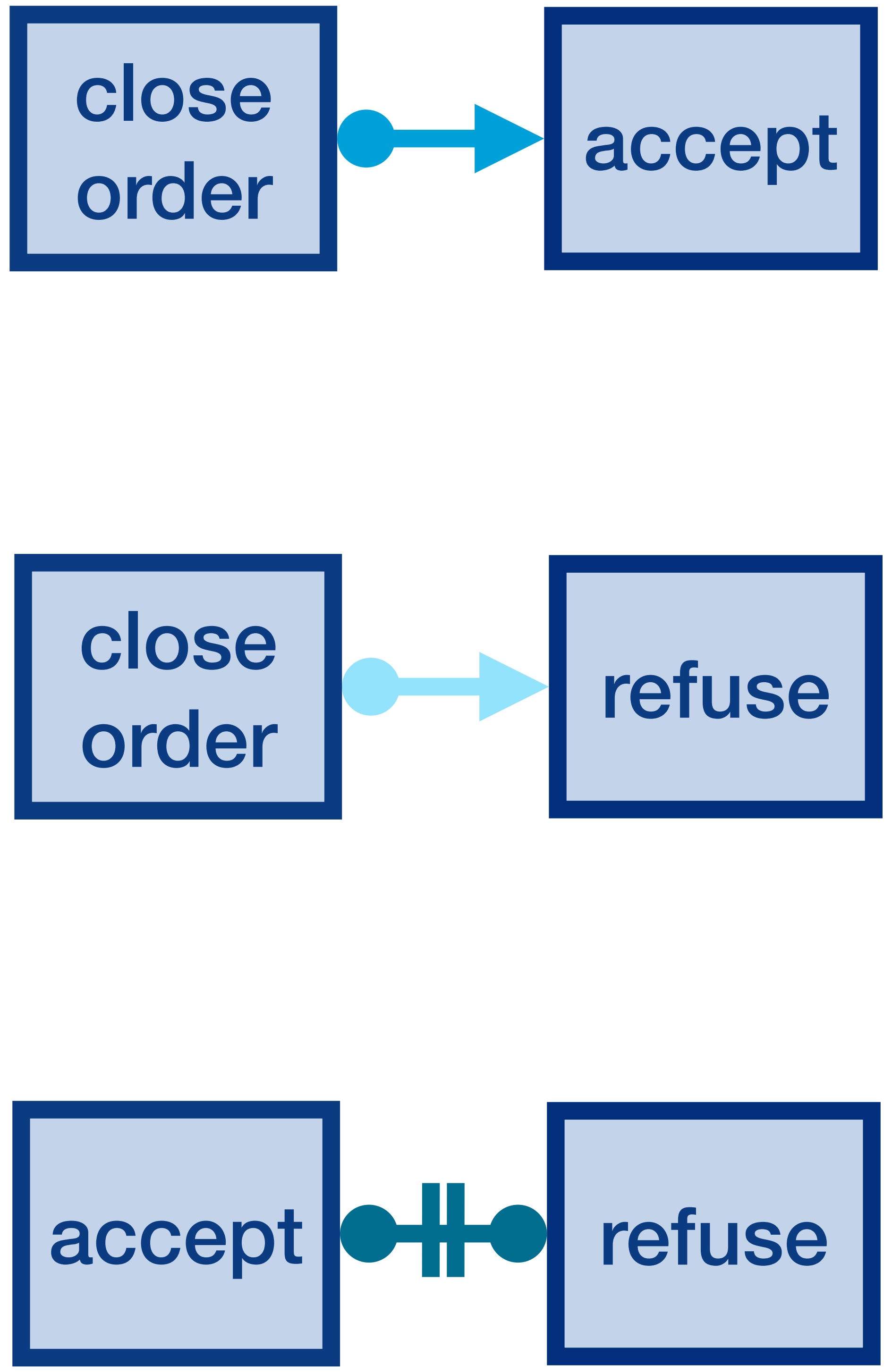




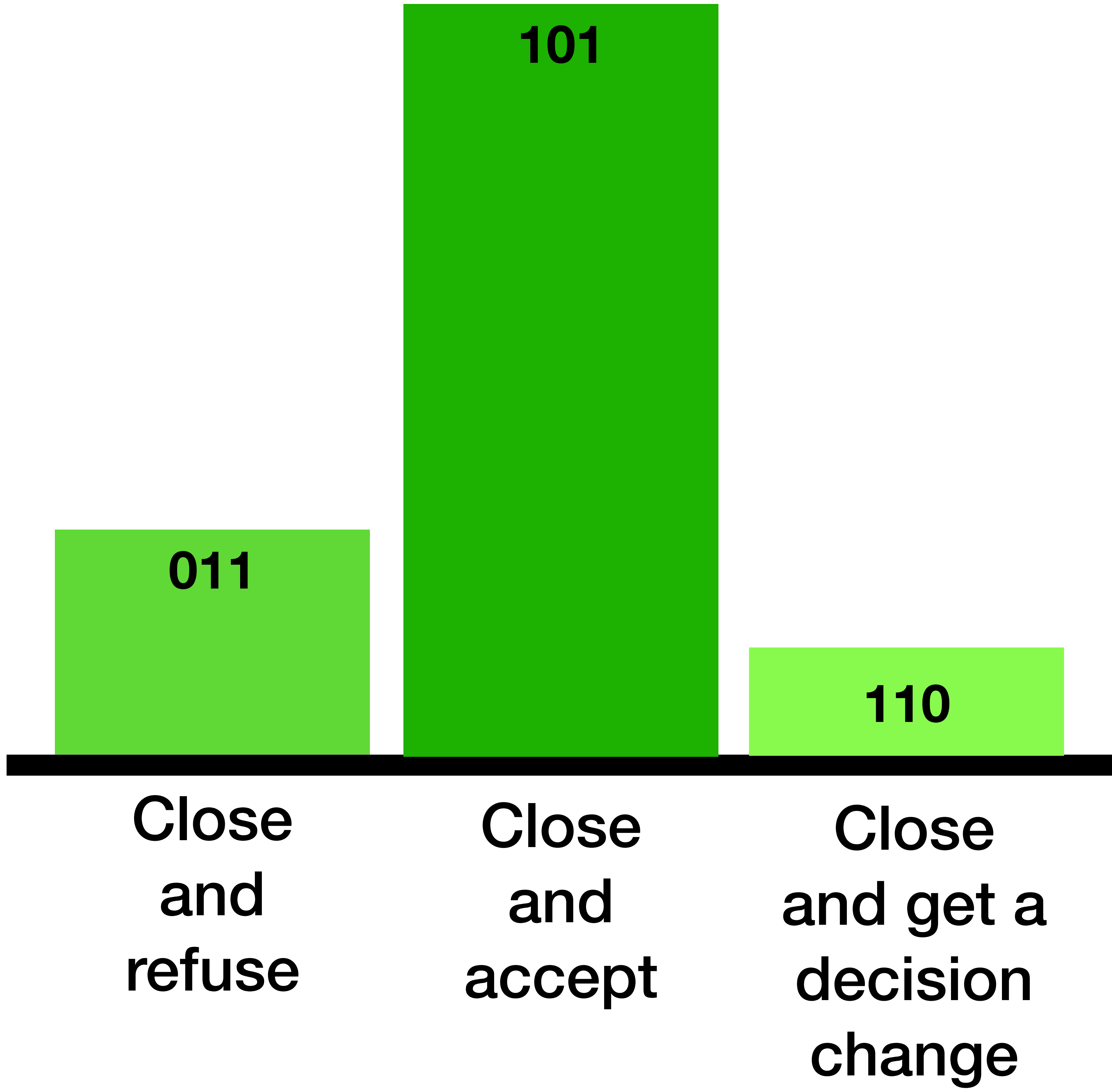
Scenarios in action

Conformance checking

<close order>

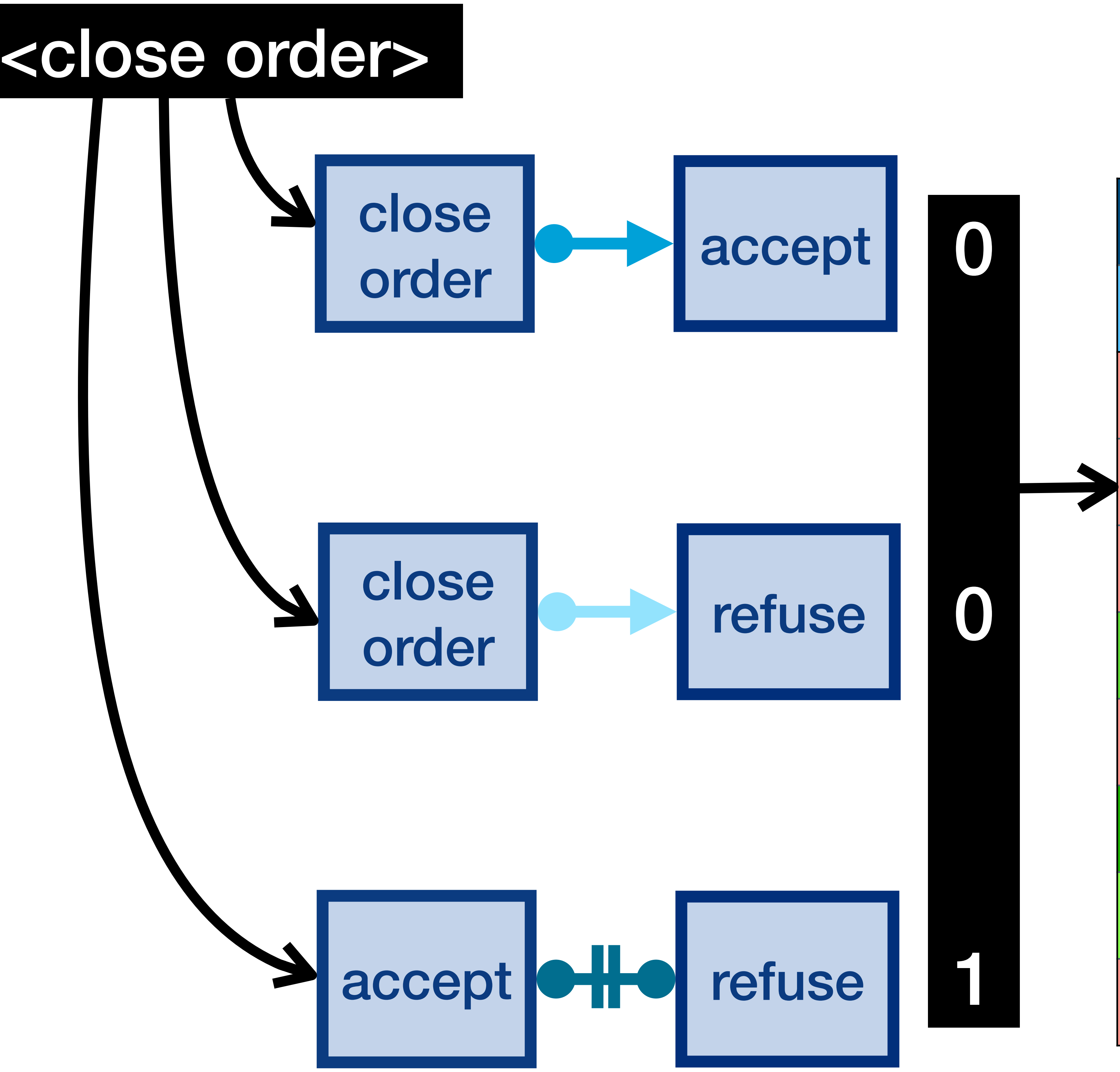


scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0



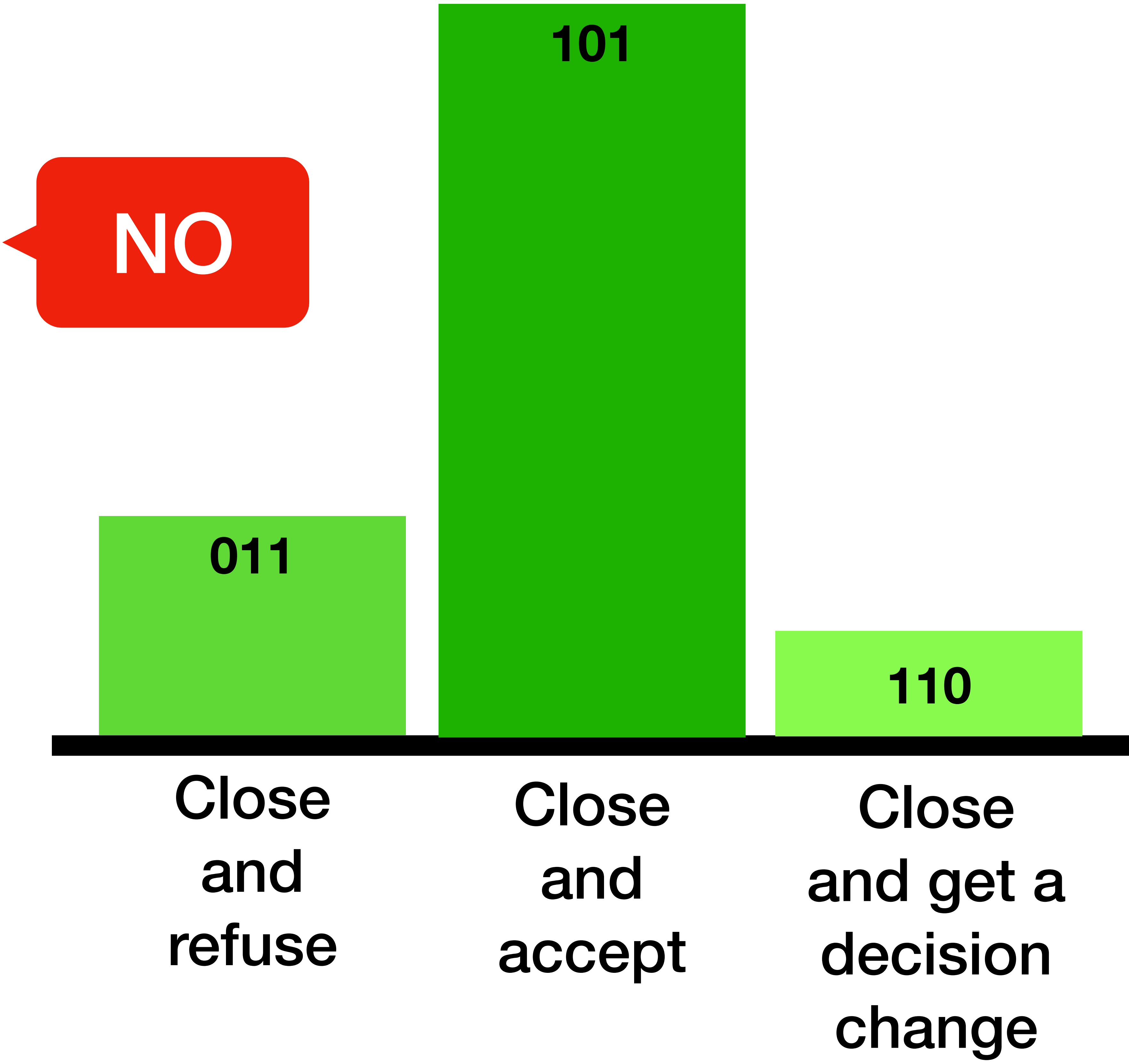
Scenarios in action

Conformance checking



scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0

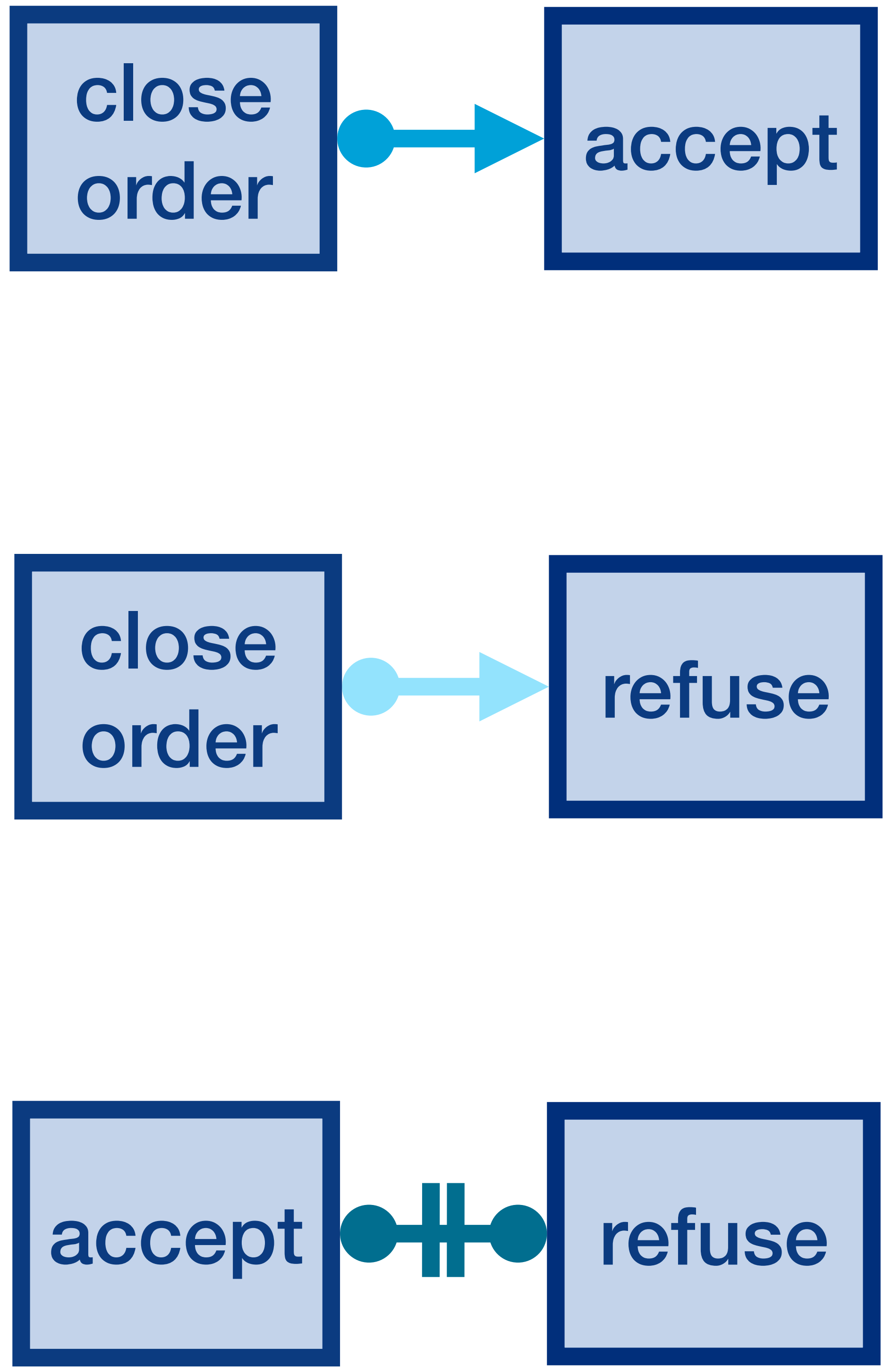
NO



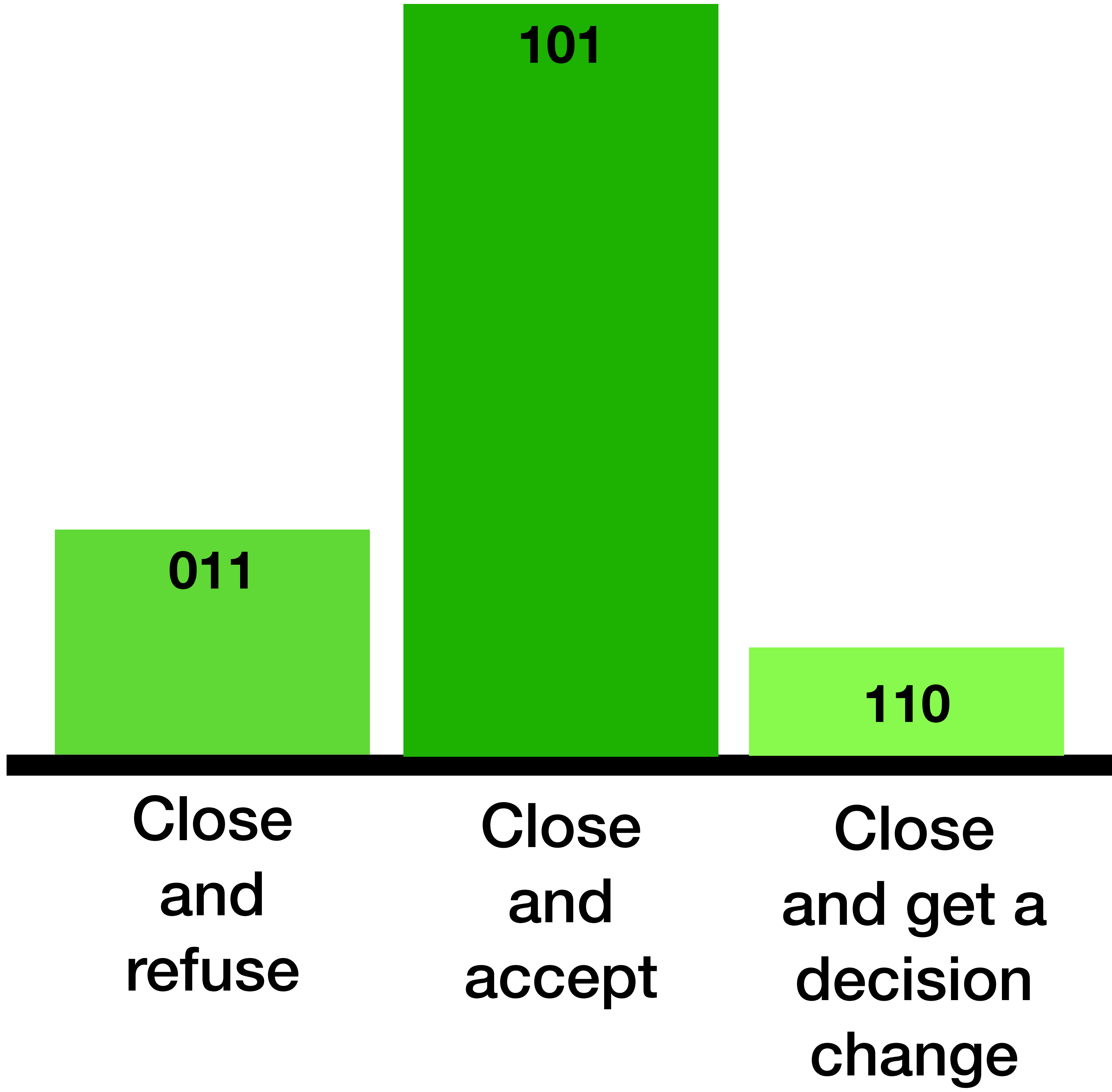
Scenarios in action

Conformance checking

<close order, accept, refuse>

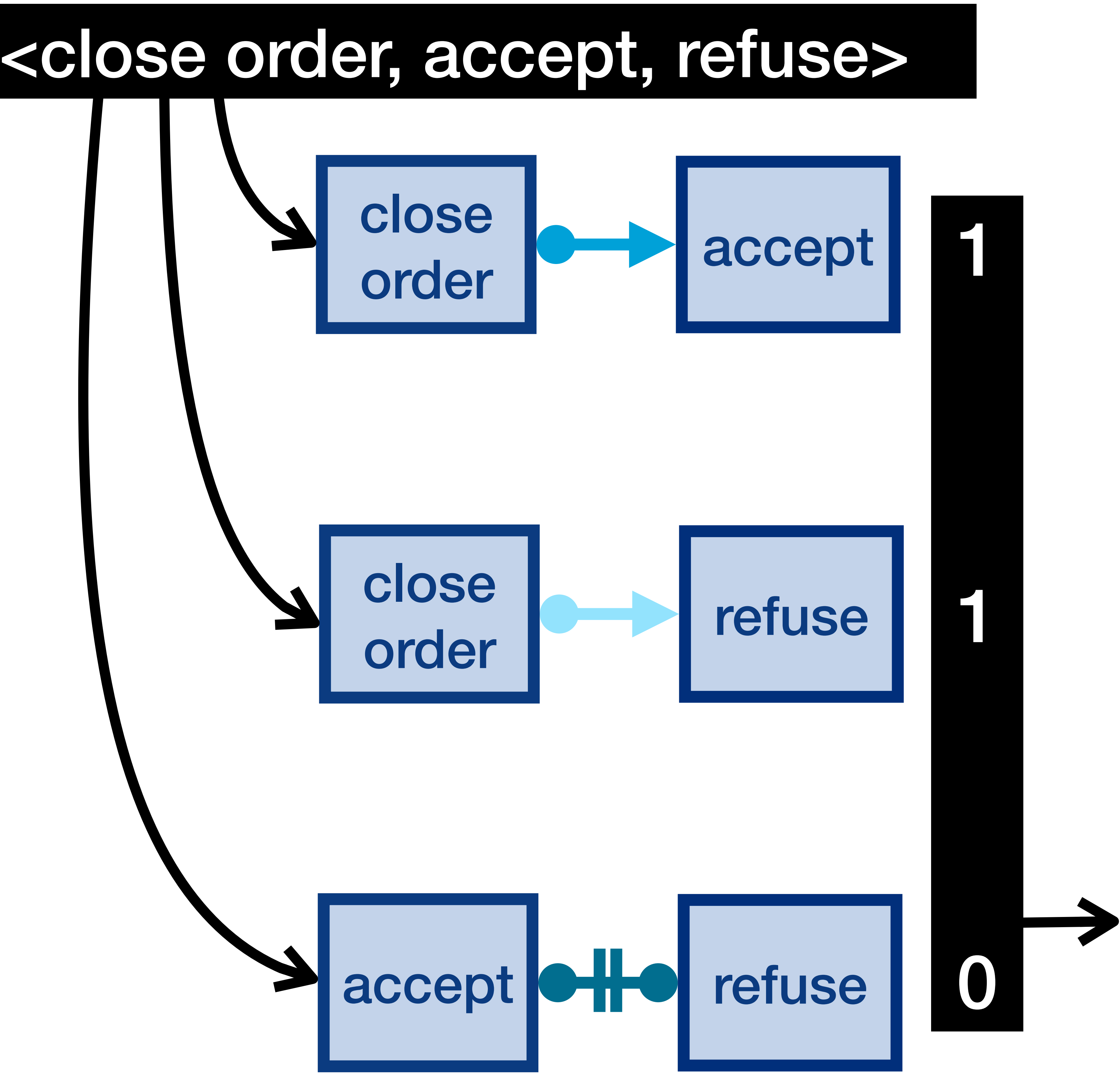


scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0

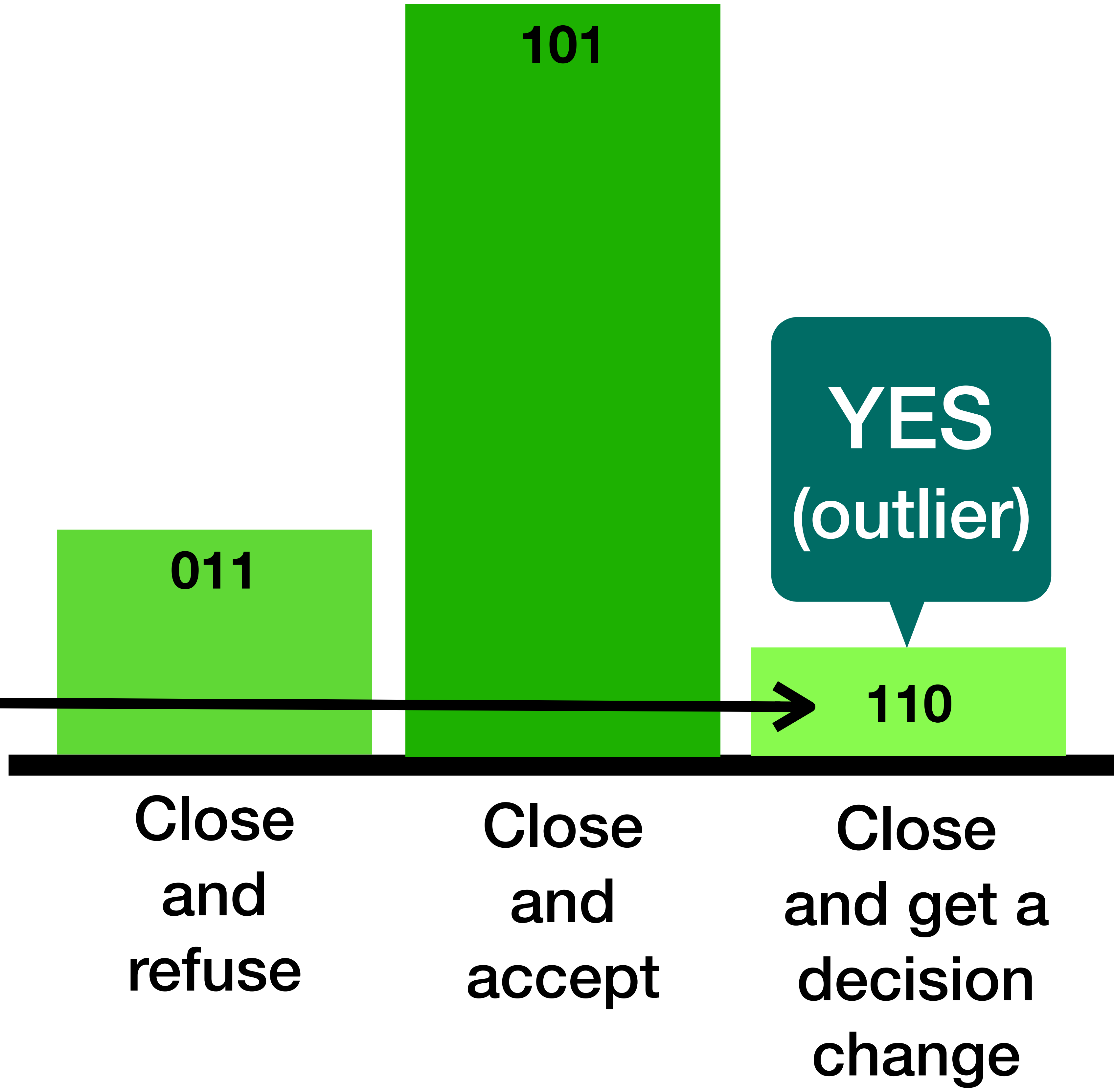


Scenarios in action

Conformance checking

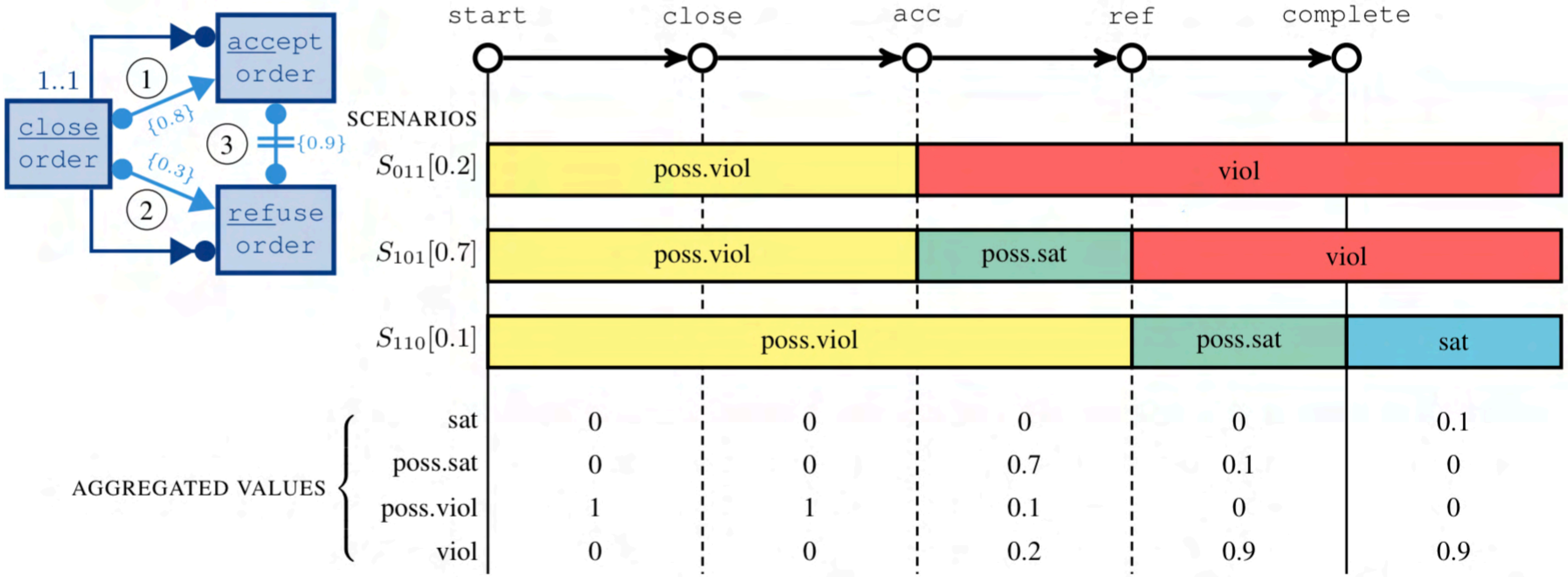


scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0



Scenarios in action

Probabilistic monitoring

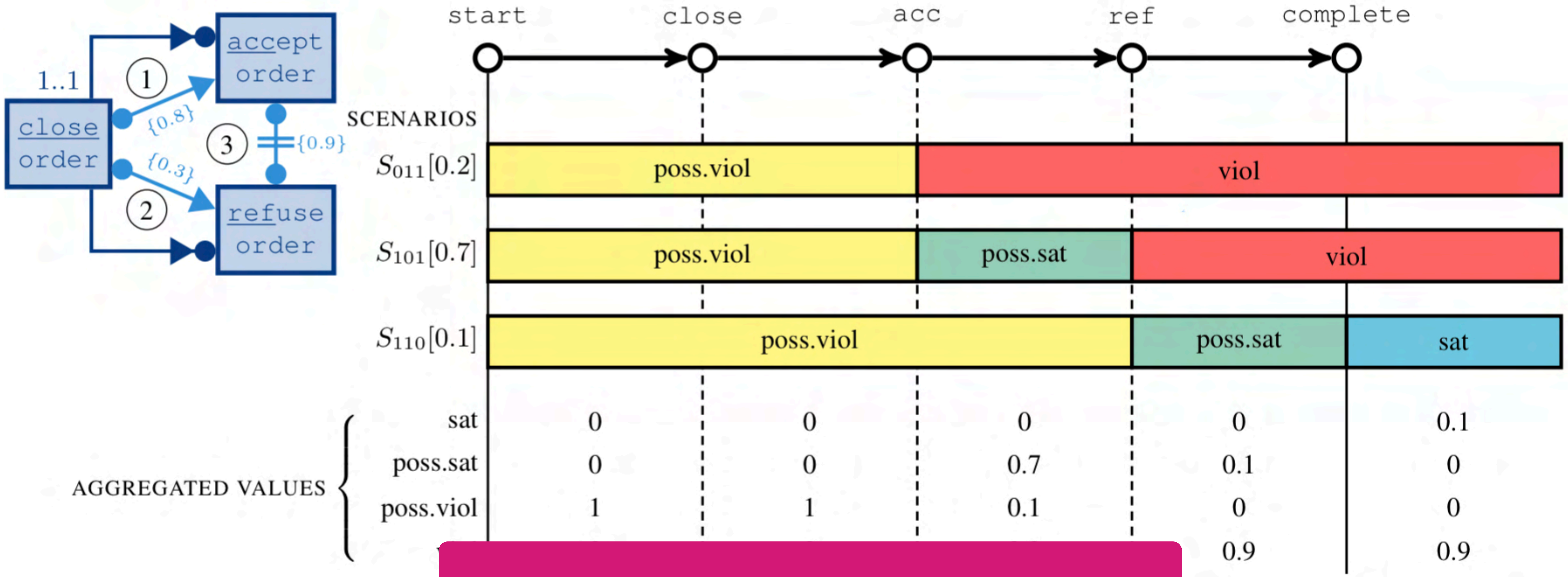


sat	0	0	0	0	0.1
poss.sat	0	0	0.7	0.1	0
poss.viol	1	1	0.1	0	0
viol	0	0	0.2	0.9	0.9

- One **global monitor per scenario**
- Monitors used in **parallel**: if multiple return the same verdict, **aggregate their probability**
- Interesting **a-priori** vs **posterior** reading of probabilities

Scenarios in action

Probabilistic monitoring



sat	0	0	0	0	0.1
poss.sat	0	0	0.7	0.1	0
poss.viol	1	1	0.1	0	0
				0.9	0.9

Human interpretability is an interesting open challenge

- One **global monitor per s**
- Monitors used in **parallel** erdict, **aggregate their probability**
- Interesting **a-priori** vs **posterior** reading of probabilities

From traces to logs

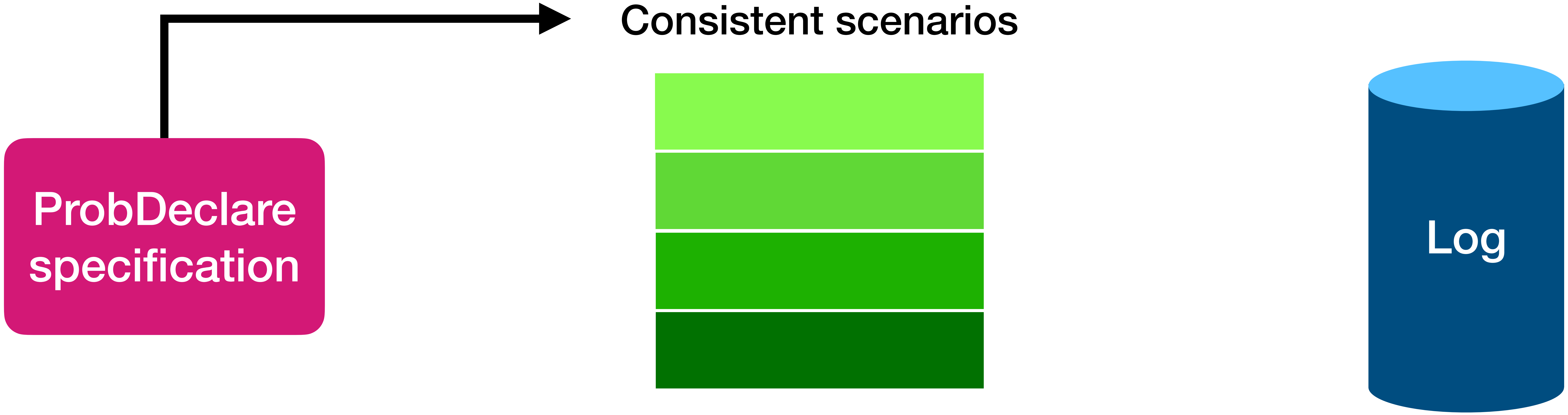
Stochastic conformance (granularity: scenario)

ProbDeclare specification

Log

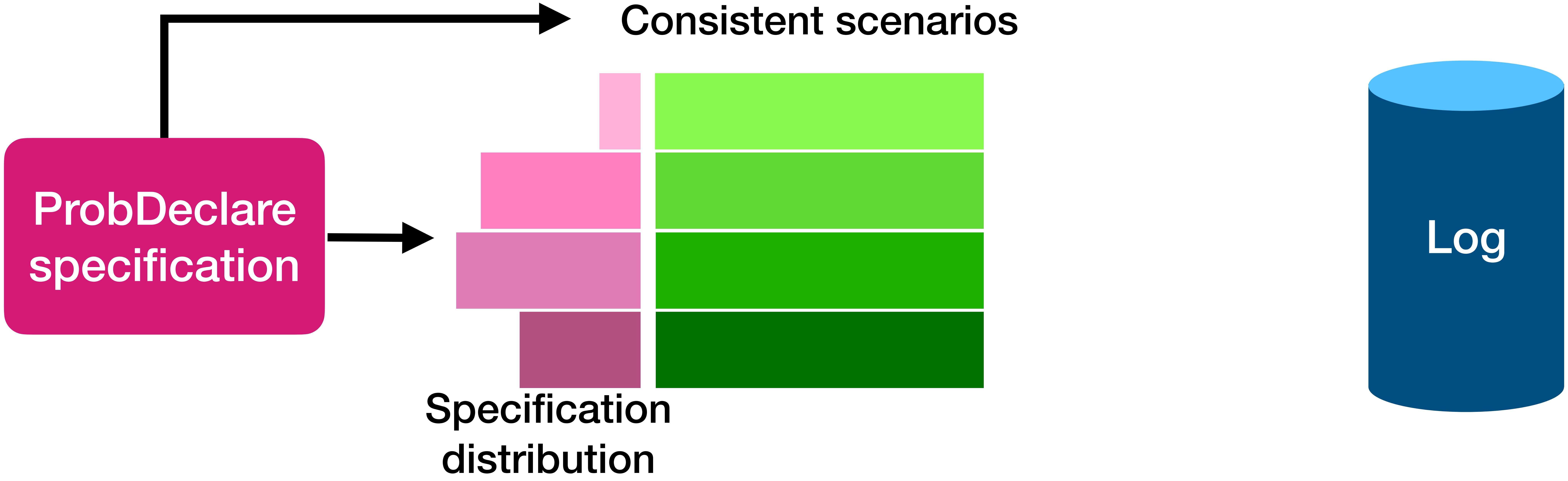
From traces to logs

Stochastic conformance (granularity: scenario)



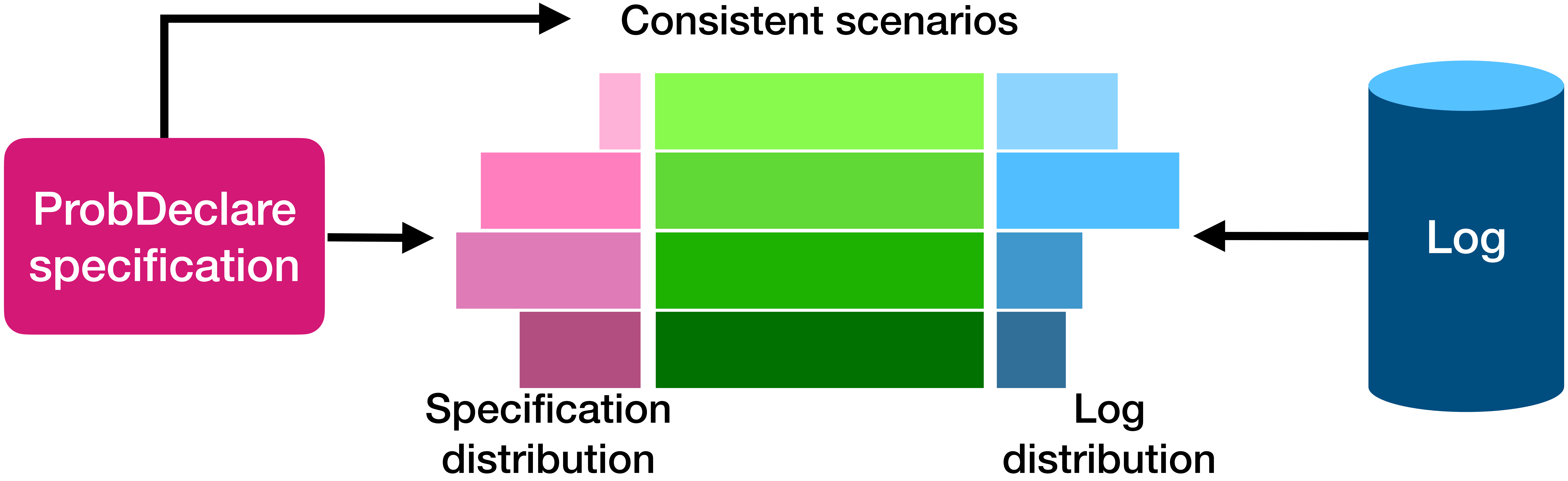
From traces to logs

Stochastic conformance (granularity: scenario)



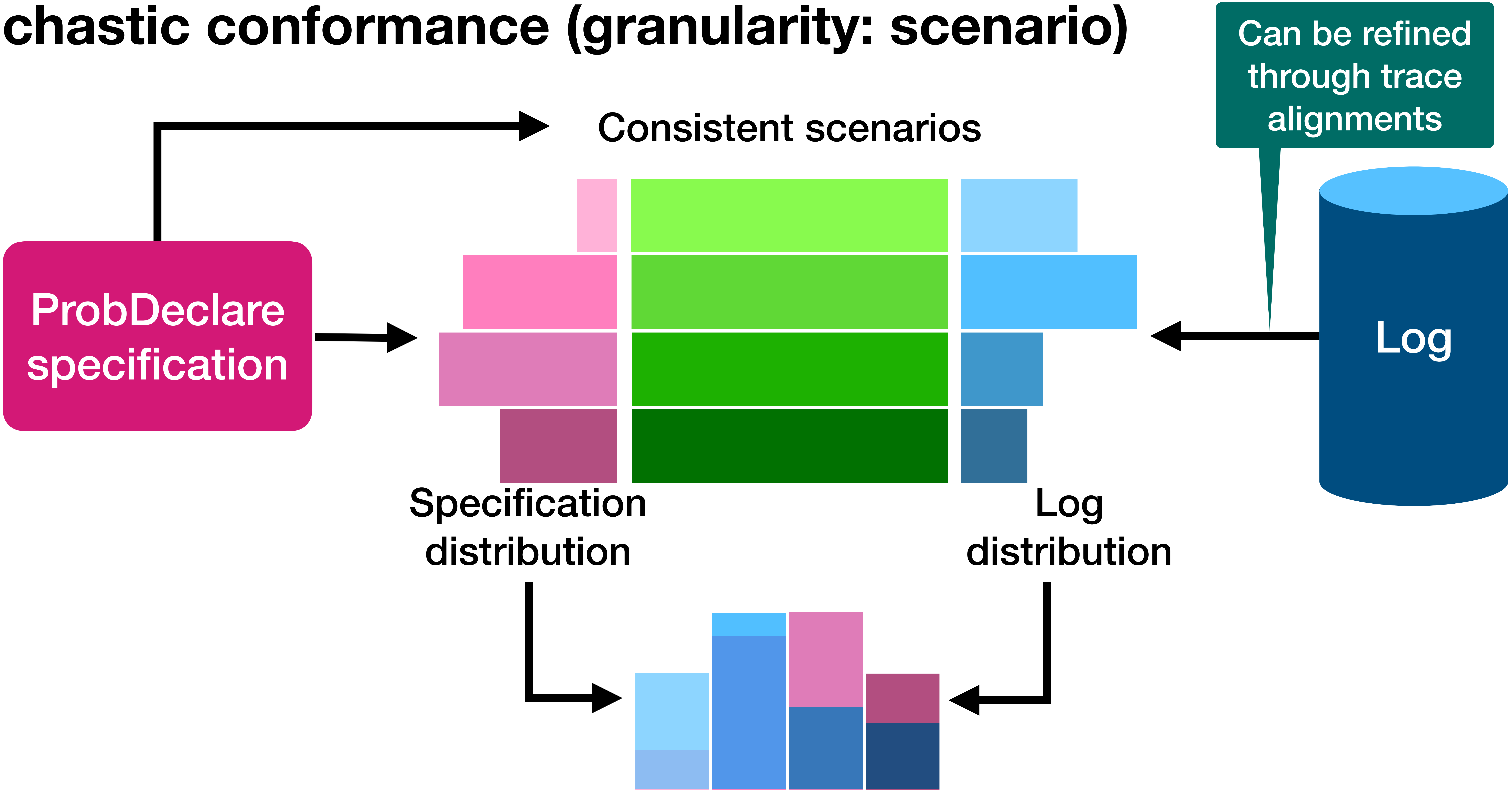
From traces to logs

Stochastic conformance (granularity: scenario)



From traces to logs

Stochastic conformance (granularity: scenario)



Can be refined through trace alignments

Consistent scenarios

ProbDeclare specification

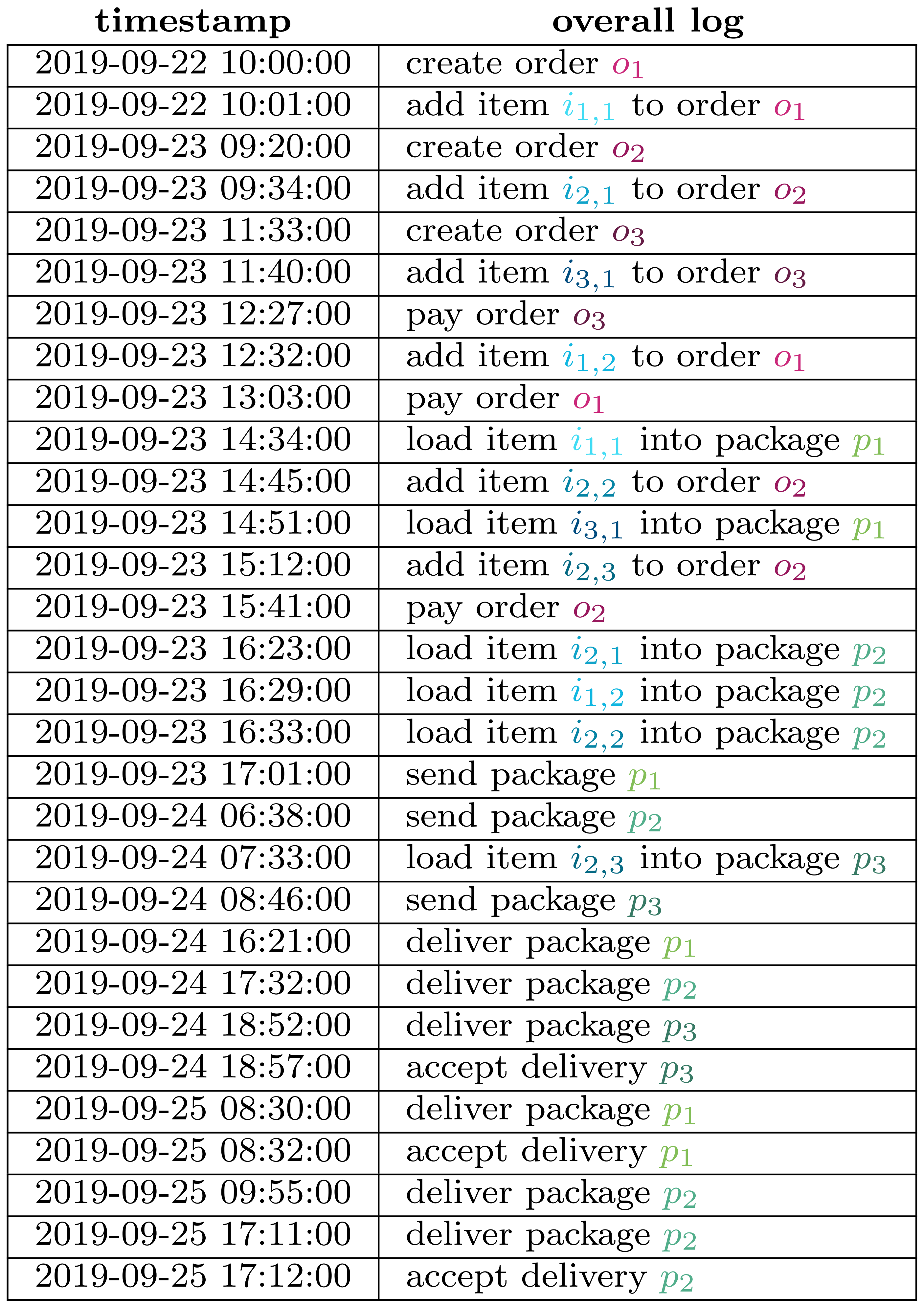
Log

Specification distribution

Log distribution

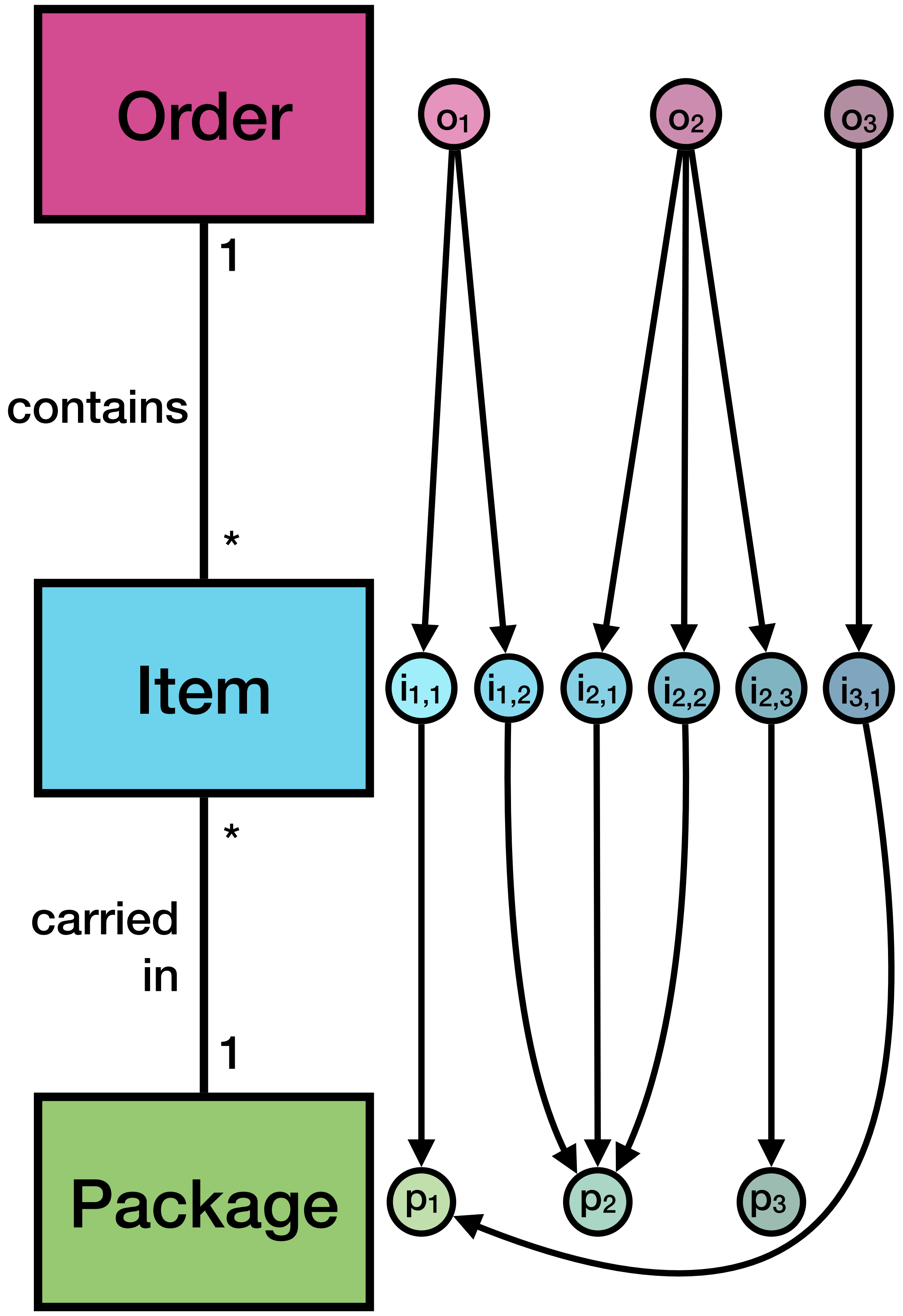
(Earth mover's) distance

Processes are not flat



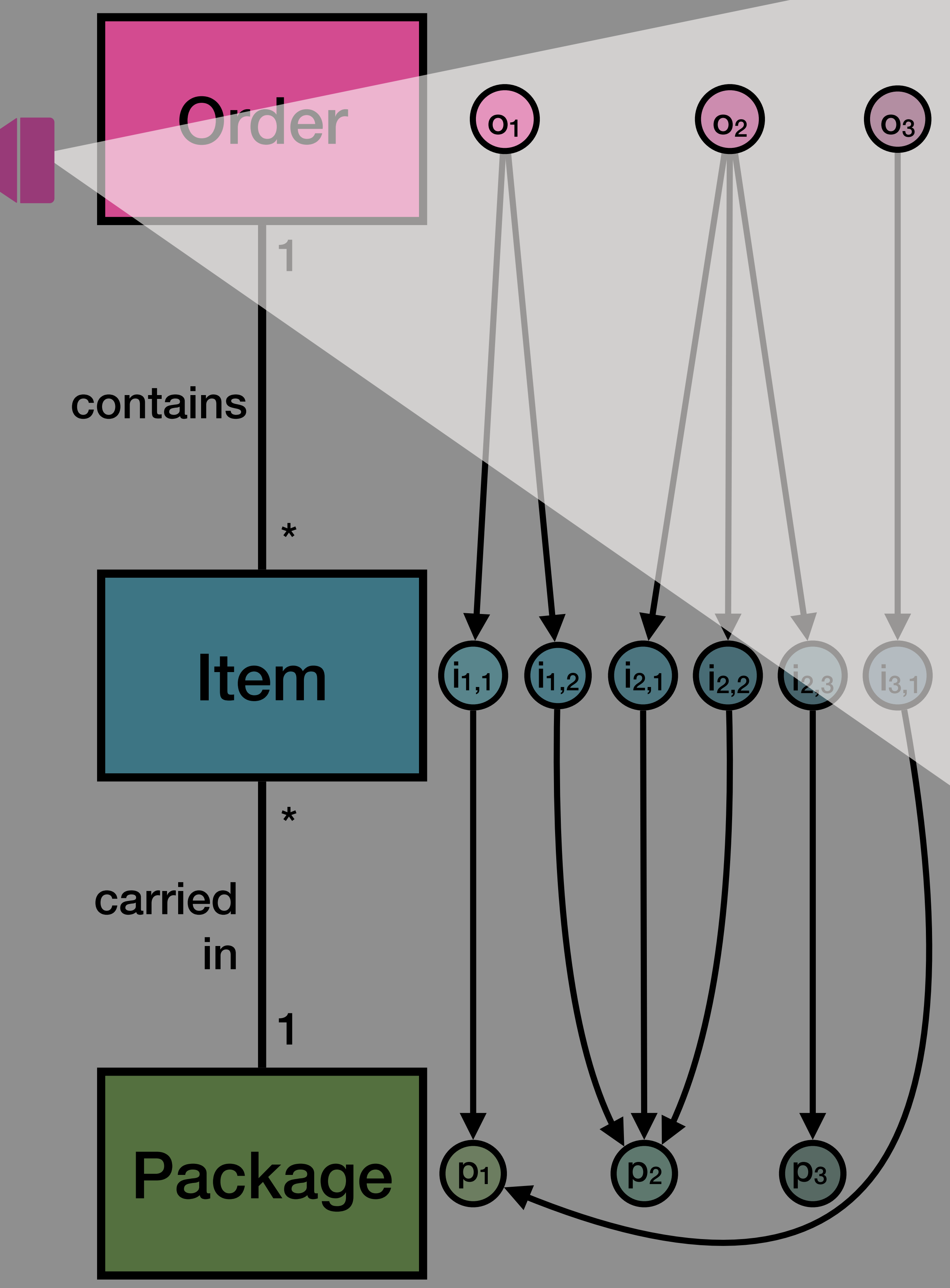
timestamp	overall log
2019-09-22 10:00:00	create order o_1
2019-09-22 10:01:00	add item $i_{1,1}$ to order o_1
2019-09-23 09:20:00	create order o_2
2019-09-23 09:34:00	add item $i_{2,1}$ to order o_2
2019-09-23 11:33:00	create order o_3
2019-09-23 11:40:00	add item $i_{3,1}$ to order o_3
2019-09-23 12:27:00	pay order o_3
2019-09-23 12:32:00	add item $i_{1,2}$ to order o_1
2019-09-23 13:03:00	pay order o_1
2019-09-23 14:34:00	load item $i_{1,1}$ into package p_1
2019-09-23 14:45:00	add item $i_{2,2}$ to order o_2
2019-09-23 14:51:00	load item $i_{3,1}$ into package p_1
2019-09-23 15:12:00	add item $i_{2,3}$ to order o_2
2019-09-23 15:41:00	pay order o_2
2019-09-23 16:23:00	load item $i_{2,1}$ into package p_2
2019-09-23 16:29:00	load item $i_{1,2}$ into package p_2
2019-09-23 16:33:00	load item $i_{2,2}$ into package p_2
2019-09-23 17:01:00	send package p_1
2019-09-24 06:38:00	send package p_2
2019-09-24 07:33:00	load item $i_{2,3}$ into package p_3
2019-09-24 08:46:00	send package p_3
2019-09-24 16:21:00	deliver package p_1
2019-09-24 17:32:00	deliver package p_2
2019-09-24 18:52:00	deliver package p_3
2019-09-24 18:57:00	accept delivery p_3
2019-09-25 08:30:00	deliver package p_1
2019-09-25 08:32:00	accept delivery p_1
2019-09-25 09:55:00	deliver package p_2
2019-09-25 17:11:00	deliver package p_2
2019-09-25 17:12:00	accept delivery p_2

Processes are not flat



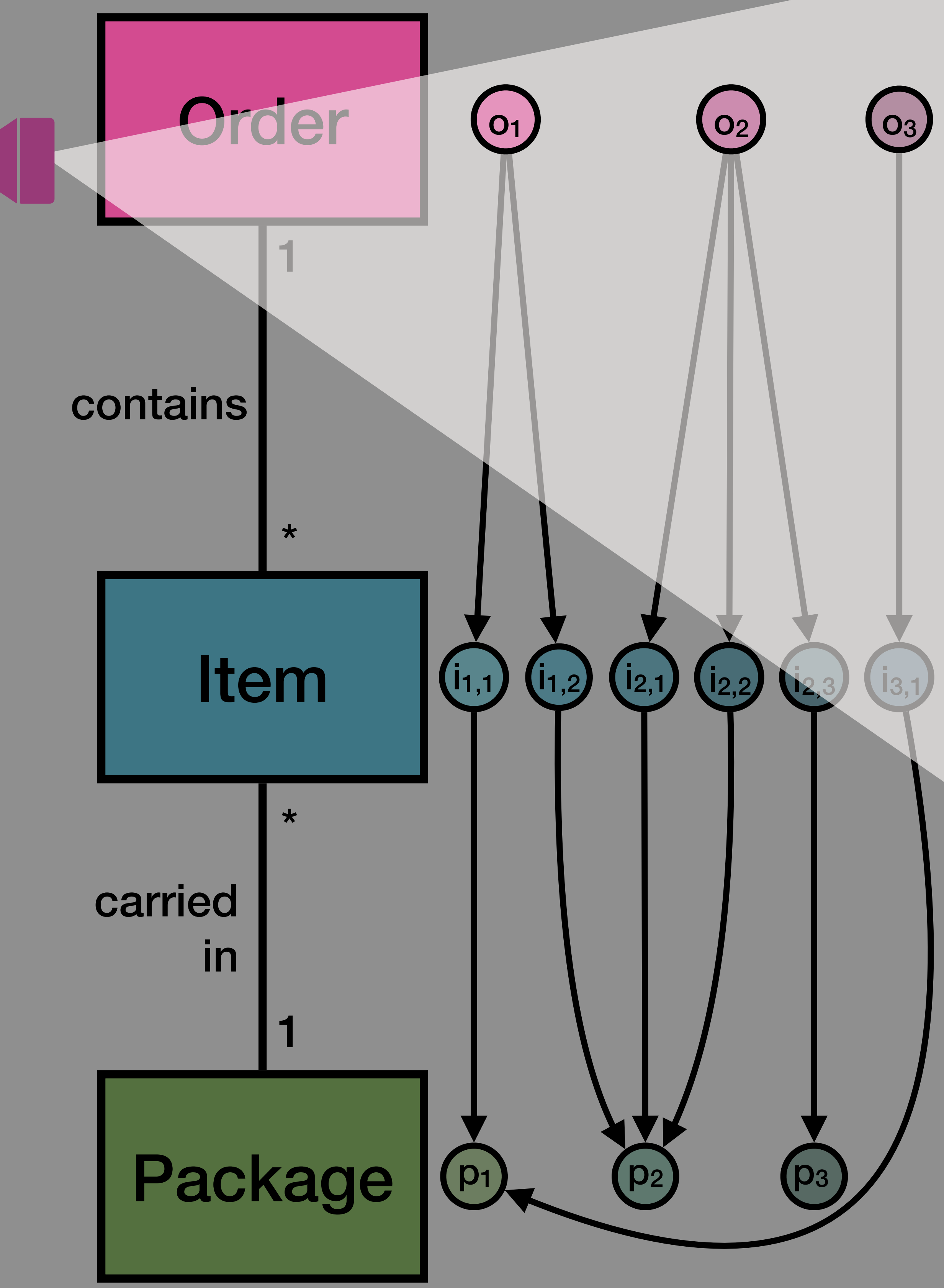
timestamp	overall log
2019-09-22 10:00:00	create order o_1
2019-09-22 10:01:00	add item $i_{1,1}$ to order o_1
2019-09-23 09:20:00	create order o_2
2019-09-23 09:34:00	add item $i_{2,1}$ to order o_2
2019-09-23 11:33:00	create order o_3
2019-09-23 11:40:00	add item $i_{3,1}$ to order o_3
2019-09-23 12:27:00	pay order o_3
2019-09-23 12:32:00	add item $i_{1,2}$ to order o_1
2019-09-23 13:03:00	pay order o_1
2019-09-23 14:34:00	load item $i_{1,1}$ into package p_1
2019-09-23 14:45:00	add item $i_{2,2}$ to order o_2
2019-09-23 14:51:00	load item $i_{3,1}$ into package p_1
2019-09-23 15:12:00	add item $i_{2,3}$ to order o_2
2019-09-23 15:41:00	pay order o_2
2019-09-23 16:23:00	load item $i_{2,1}$ into package p_2
2019-09-23 16:29:00	load item $i_{1,2}$ into package p_2
2019-09-23 16:33:00	load item $i_{2,2}$ into package p_2
2019-09-23 17:01:00	send package p_1
2019-09-24 06:38:00	send package p_2
2019-09-24 07:33:00	load item $i_{2,3}$ into package p_3
2019-09-24 08:46:00	send package p_3
2019-09-24 16:21:00	deliver package p_1
2019-09-24 17:32:00	deliver package p_2
2019-09-24 18:52:00	deliver package p_3
2019-09-24 18:57:00	accept delivery p_3
2019-09-25 08:30:00	deliver package p_1
2019-09-25 08:32:00	accept delivery p_1
2019-09-25 09:55:00	deliver package p_2
2019-09-25 17:11:00	deliver package p_2
2019-09-25 17:12:00	accept delivery p_2

Processes are not flat

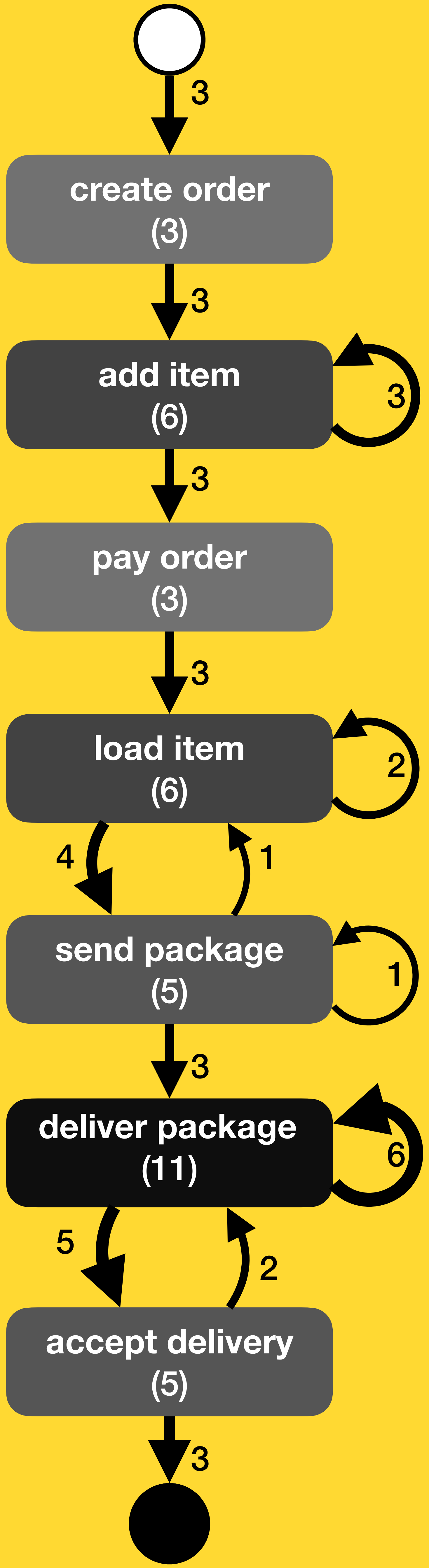


			event log for orders	
timestamp	overall log	order o_1	order o_2	order o_3
2019-09-22 10:00:00	create order o_1	create order		
2019-09-22 10:01:00	add item $i_{1,1}$ to order o_1	add item		
2019-09-23 09:20:00	create order o_2		create order	
2019-09-23 09:34:00	add item $i_{2,1}$ to order o_2		add item	
2019-09-23 11:33:00	create order o_3			create order
2019-09-23 11:40:00	add item $i_{3,1}$ to order o_3			add item
2019-09-23 12:27:00	pay order o_3			pay order
2019-09-23 12:32:00	add item $i_{1,2}$ to order o_1	add item		
2019-09-23 13:03:00	pay order o_1	pay order		
2019-09-23 14:34:00	load item $i_{1,1}$ into package p_1	load item		
2019-09-23 14:45:00	add item $i_{2,2}$ to order o_2		add item	
2019-09-23 14:51:00	load item $i_{3,1}$ into package p_1			load item
2019-09-23 15:12:00	add item $i_{2,3}$ to order o_2		add item	
2019-09-23 15:41:00	pay order o_2		pay order	
2019-09-23 16:23:00	load item $i_{2,1}$ into package p_2		load item	
2019-09-23 16:29:00	load item $i_{1,2}$ into package p_2	load item		
2019-09-23 16:33:00	load item $i_{2,2}$ into package p_2		load item	
2019-09-23 17:01:00	send package p_1	send package		send package
2019-09-24 06:38:00	send package p_2	send package	send package	
2019-09-24 07:33:00	load item $i_{2,3}$ into package p_3			load item
2019-09-24 08:46:00	send package p_3		send package	
2019-09-24 16:21:00	deliver package p_1	deliver package		deliver package
2019-09-24 17:32:00	deliver package p_2	deliver package	deliver package	
2019-09-24 18:52:00	deliver package p_3		deliver package	
2019-09-24 18:57:00	accept delivery p_3		accept delivery	
2019-09-25 08:30:00	deliver package p_1	deliver package		deliver package
2019-09-25 08:32:00	accept delivery p_1	accept delivery		accept delivery
2019-09-25 09:55:00	deliver package p_2	deliver package	deliver package	
2019-09-25 17:11:00	deliver package p_2	deliver package	deliver package	
2019-09-25 17:12:00	accept delivery p_2	accept delivery	accept delivery	

Processes are not flat

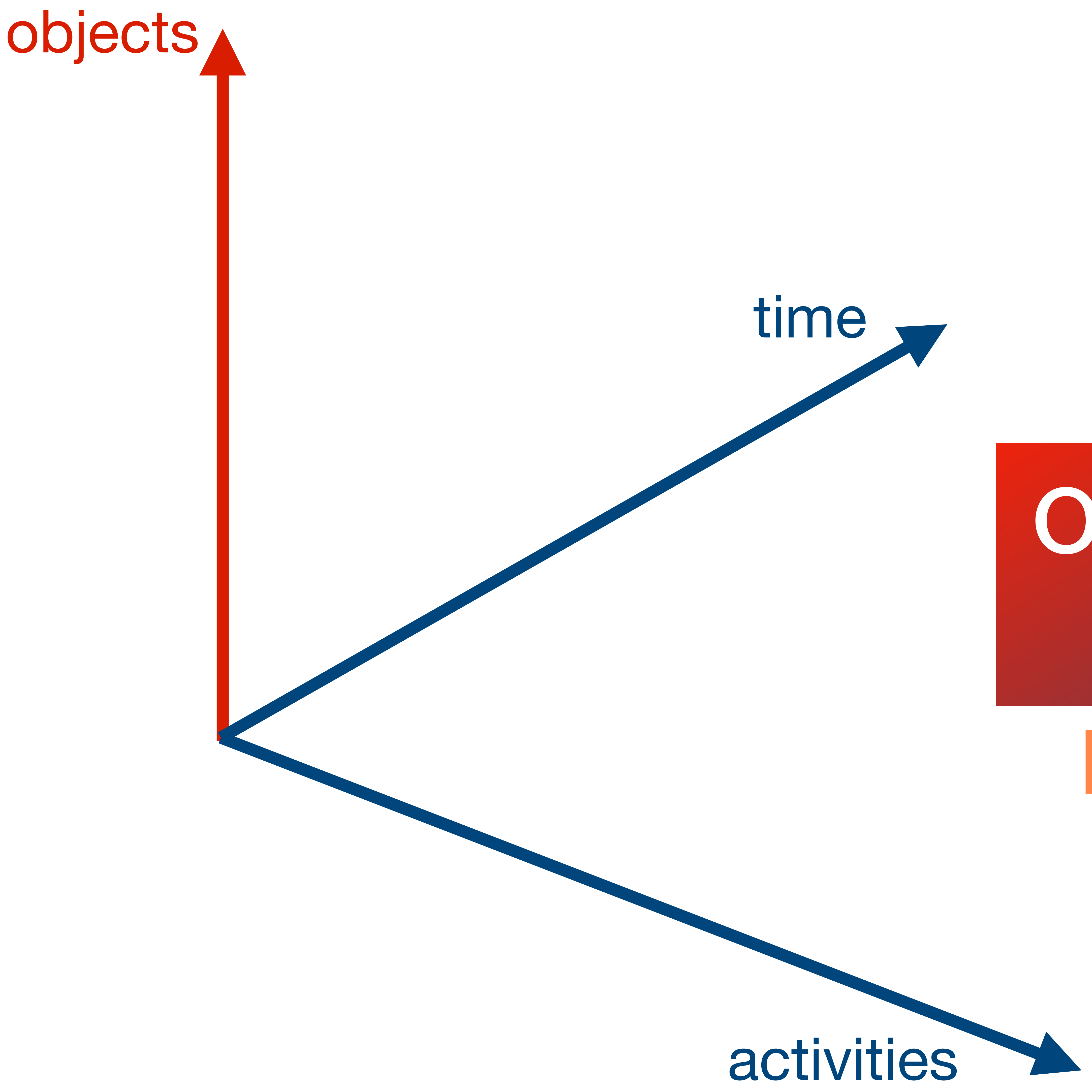


timestamp	overall log		event log for orders	
		order o_1	order o_2	order o_3
2019-09-22 10:00:00	create order o_1	create order		
2019-09-22 10:01:00	add item $i_{1,1}$ to order o_1	add item		
2019-09-23 09:20:00	create order o_2		create order	
2019-09-23 09:34:00	add item $i_{2,1}$ to order o_2		add item	
2019-09-23 11:33:00	create order o_3			create order
2019-09-23 11:40:00	add item $i_{3,1}$ to order o_3			add item
2019-09-23 12:27:00	pay order o_3			pay order
2019-09-23 12:32:00	add item $i_{1,2}$ to order o_1	add item		
2019-09-23 13:03:00	pay order o_1	pay order		
2019-09-23 14:34:00	load item $i_{1,1}$ into package p_1	load item		
2019-09-23 14:45:00	add item $i_{2,2}$ to order o_2		add item	
2019-09-23 14:51:00	load item $i_{3,1}$ into package p_1			load item
2019-09-23 15:12:00	add item $i_{2,3}$ to order o_2		add item	
2019-09-23 15:41:00	pay order o_2		pay order	
2019-09-23 16:23:00	load item $i_{2,1}$ into package p_2		load item	
2019-09-23 16:29:00	load item $i_{1,2}$ into package p_2	load item		
2019-09-23 16:33:00	load item $i_{2,2}$ into package p_2		load item	
2019-09-23 17:01:00	send package p_1	send package		send package
2019-09-24 06:38:00	send package p_2	send package	send package	
2019-09-24 07:33:00	load item $i_{2,3}$ into package p_3			load item
2019-09-24 08:46:00	send package p_3		send package	
2019-09-24 16:21:00	deliver package p_1	deliver package		deliver package
2019-09-24 17:32:00	deliver package p_2	deliver package	deliver package	
2019-09-24 18:52:00	deliver package p_3		deliver package	
2019-09-24 18:57:00	accept delivery p_3		accept delivery	
2019-09-25 08:30:00	deliver package p_1	deliver package		deliver package
2019-09-25 08:32:00	accept delivery p_1	accept delivery		accept delivery
2019-09-25 09:55:00	deliver package p_2	deliver package	deliver package	
2019-09-25 17:11:00	deliver package p_2	deliver package	deliver package	
2019-09-25 17:12:00	accept delivery p_2	accept delivery	accept delivery	



Dealing with multiple objects

Need of a 3D model



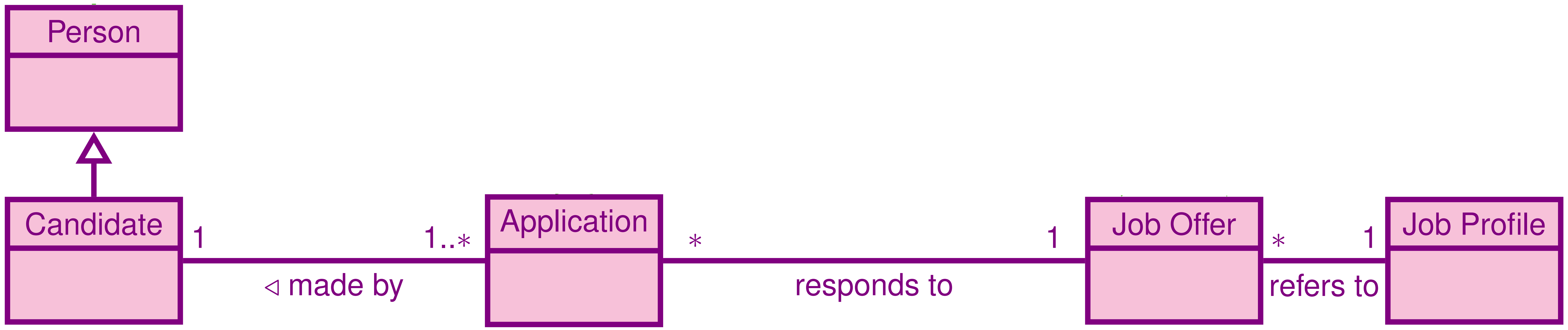
Object-centric behavioral constraints

[____,DL2017] [____,BPM2019]

Object-centric behavioral constraints

Dimension 1: data model to classify and relate objects

- classes
- relationship types
- multiplicities (one-to-one, one-to-many, many-to-many)

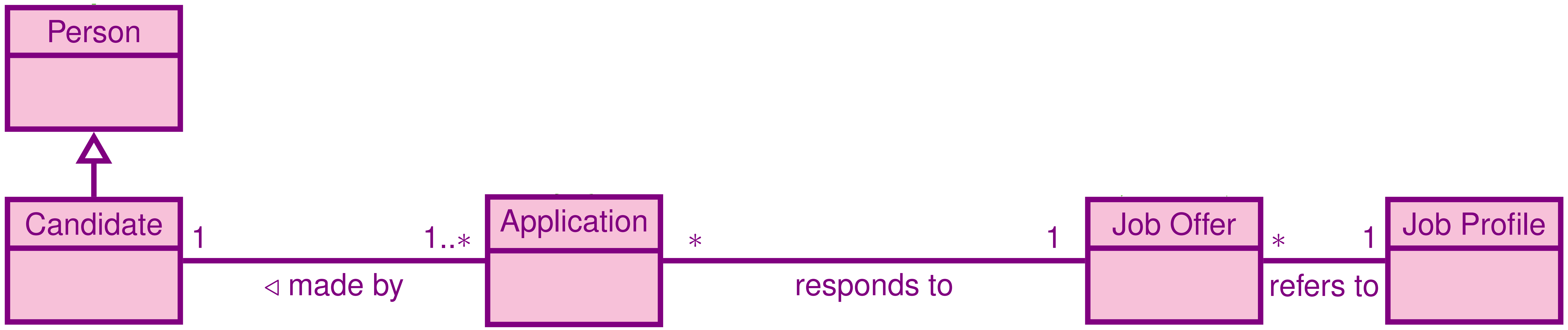


Object-centric behavioral constraints

Dimension 2: activities

- **activities**
- **activity-class relationship types**
- **multiplicities**

The **register data** task *is about* a Person.
A Job Offer is *created* by executing the **post offer** task.
A Job Offer is *closed* by **determining** the **winner**.
A Job Offer is *stopped* by **canceling** the **hiring**.
An Application is *created* by executing the **submit** task.
An Application is *promoted* by **marking** it **as eligible**.

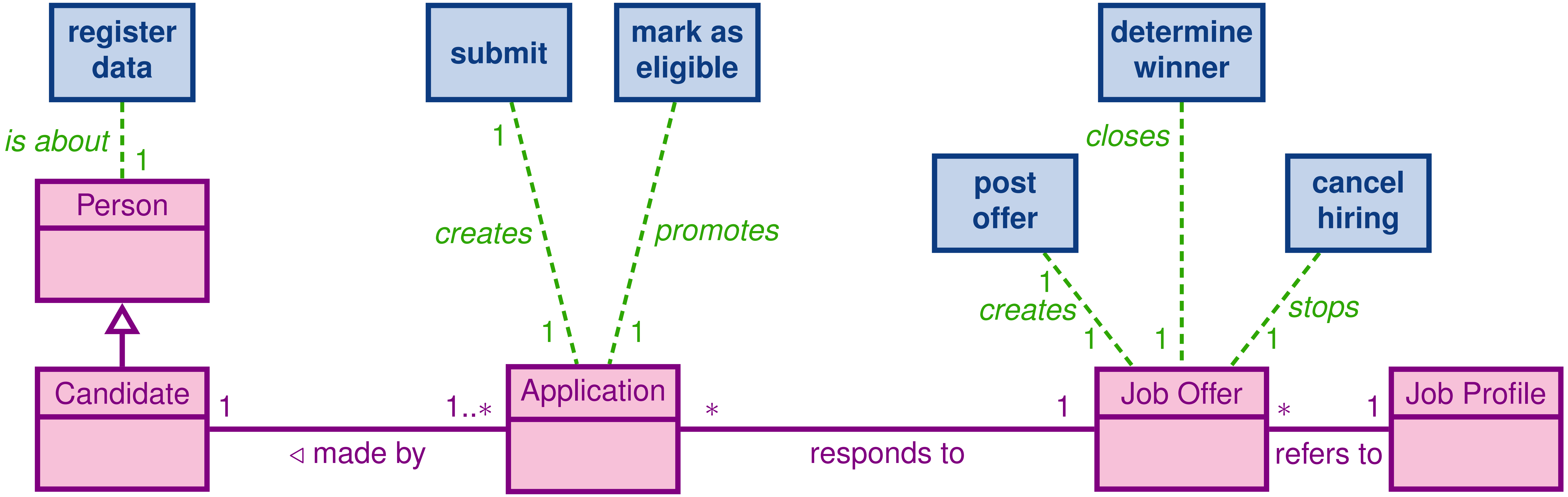


Object-centric behavioral constraints

Dimension 2: activities

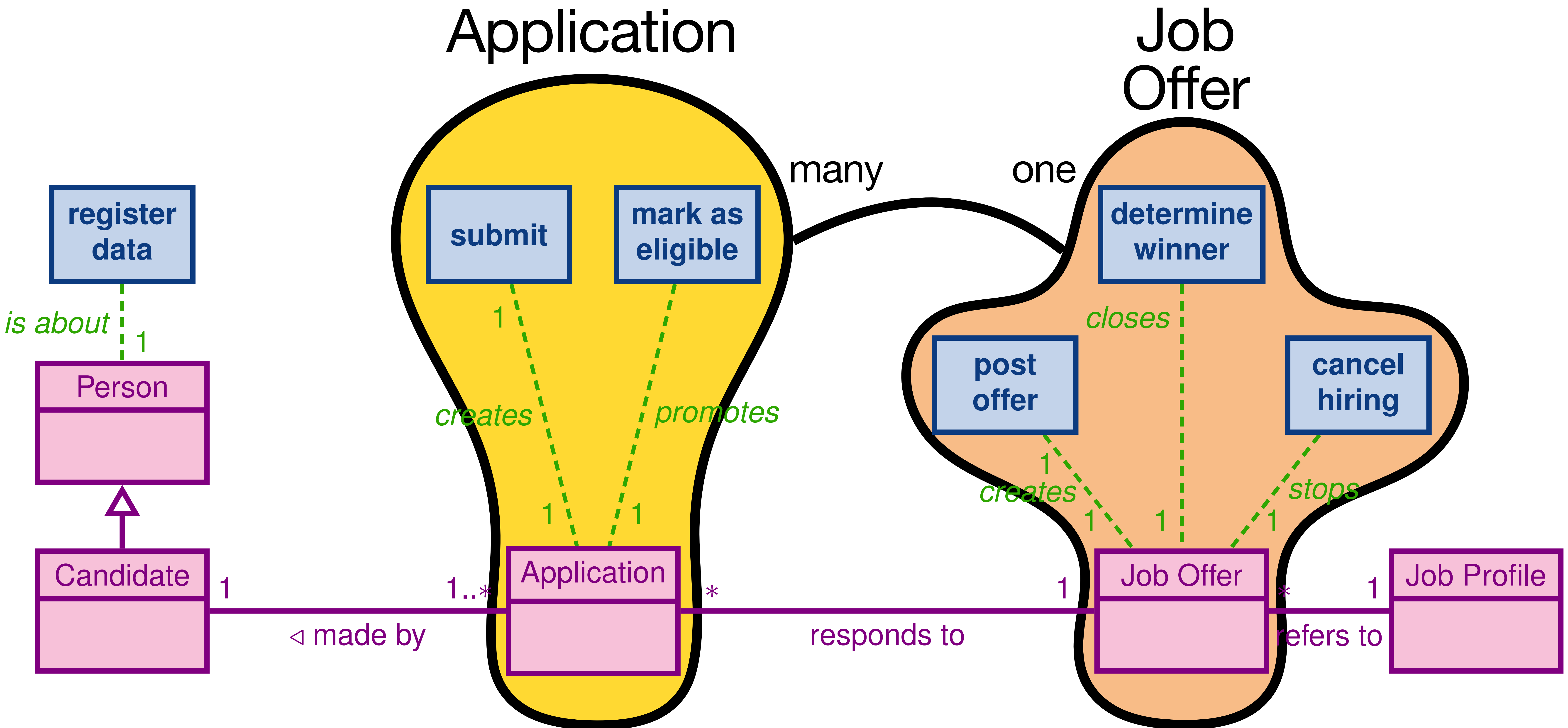
- **activities**
- **activity-class relationship types**
- **multiplicities**

The **register data** task *is about* a Person.
A Job Offer is *created* by executing the **post offer** task.
A Job Offer is *closed* by **determining** the **winner**.
A Job Offer is *stopped* by **canceling** the **hiring**.
An Application is *created* by executing the **submit** task.
An Application is *promoted* by **marking** it **as eligible**.



Object-centric behavioral constraints

Emergent object lifecycles



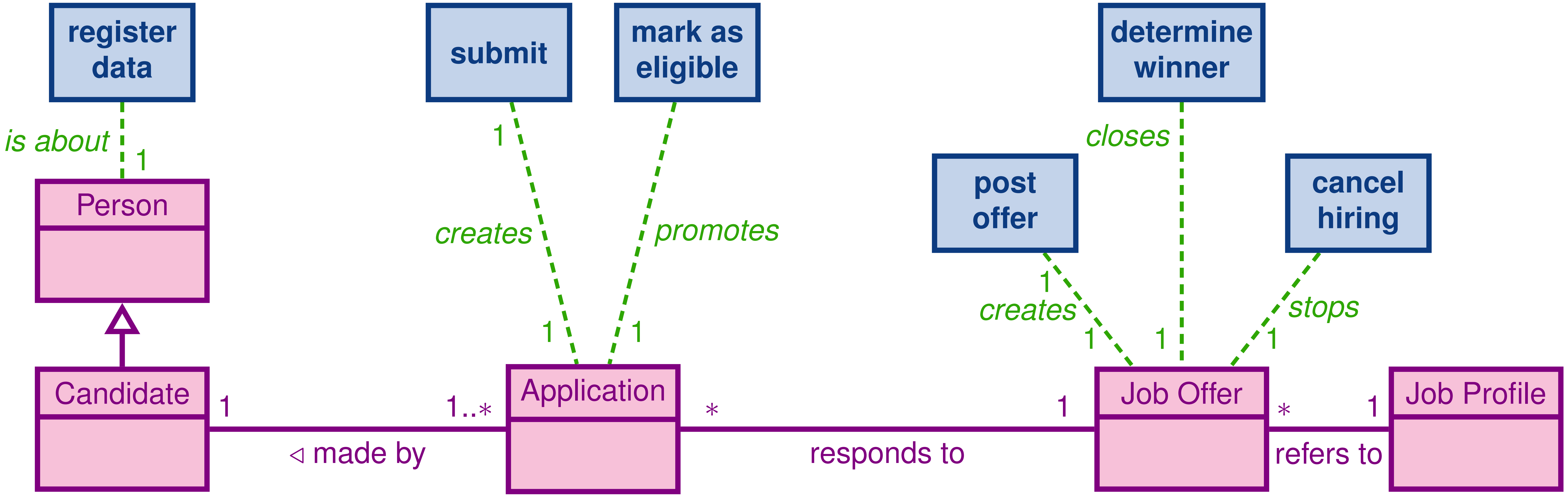
Object-centric behavioral constraints

Dimension 3: the process

- **constraints…**

A Job Offer *closed by* a **determine winner** task *cannot* be *stopped* by executing the **cancel hiring** task (and vice-versa).

An Application can be **submitted** only if, *beforehand*, the **data** *about* the Candidate who made *that* Application have been **registered**.



Object-centric behavioral constraints

Dimension 3: the process

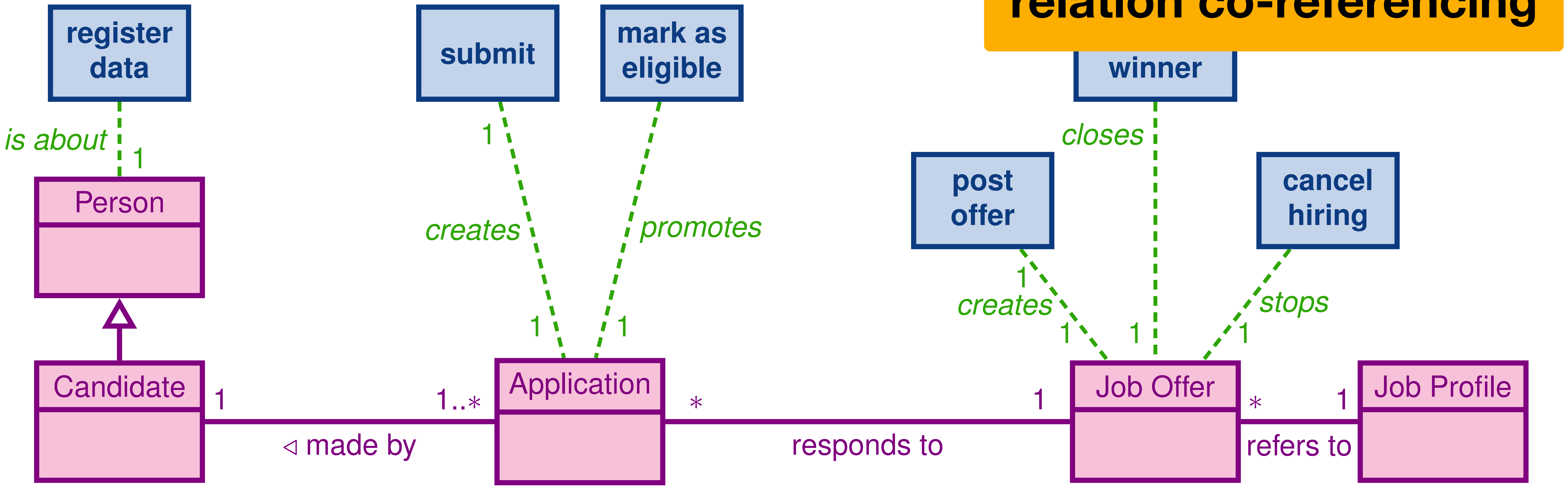
- **constraints…**
- **…with data co-referencing**

object co-referencing

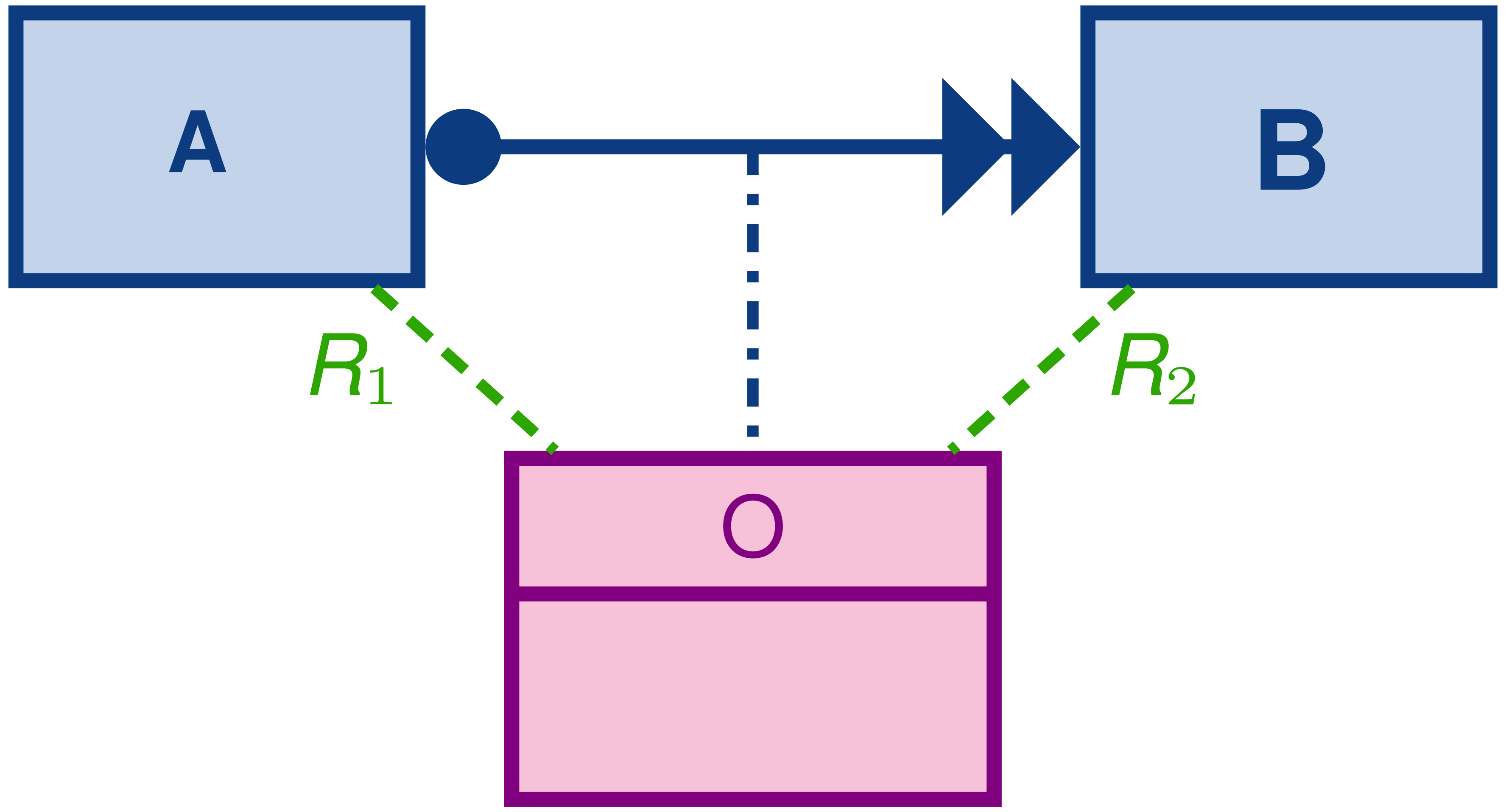
A Job Offer *closed by* a **determine winner** task *cannot* be *stopped* by executing the **cancel hiring** task (and vice-versa).

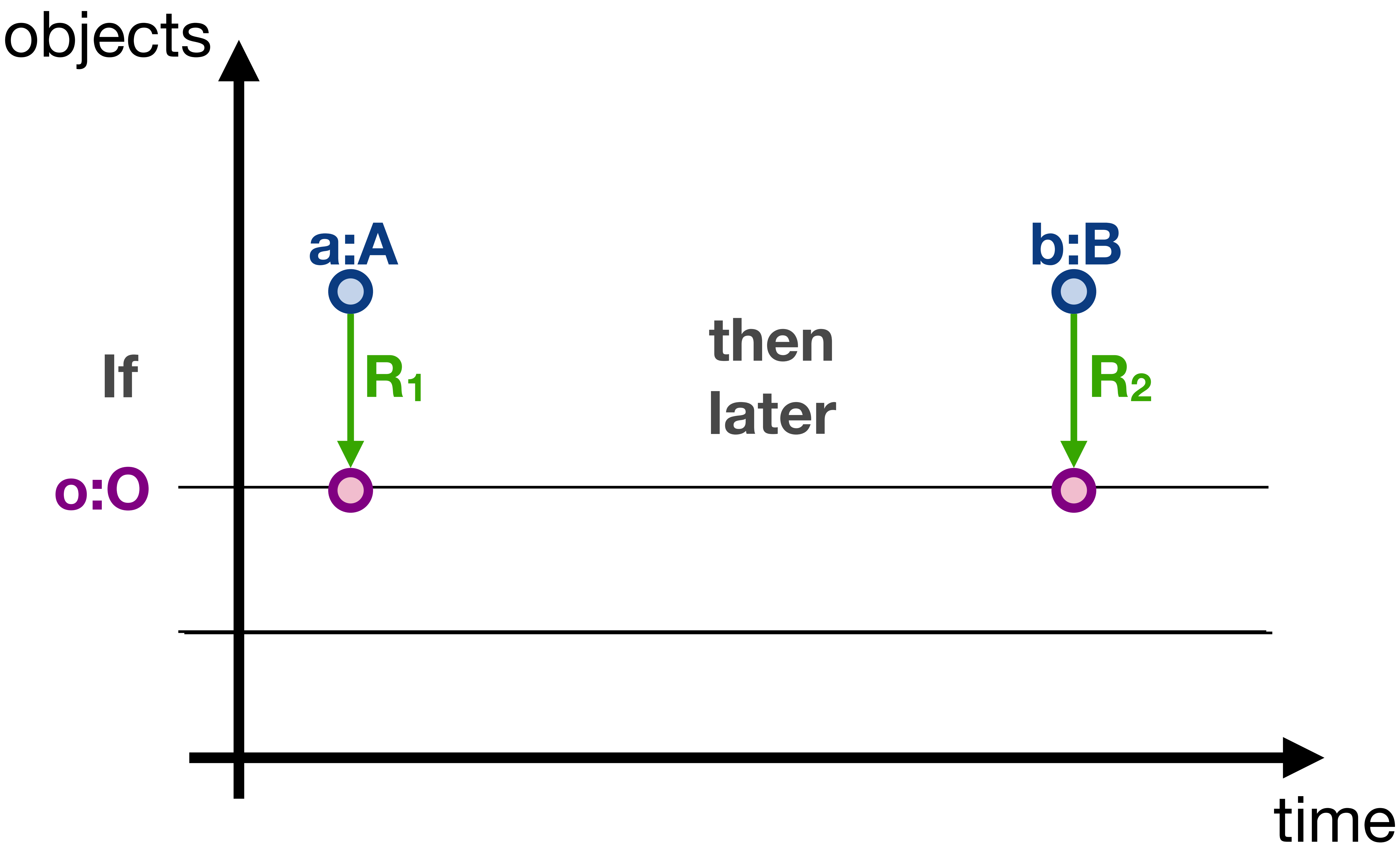
An Application can be **submitted** only if, *beforehand*, the **data** *about* the Candidate who made *that* Application have been **registered**.

relation co-referencing

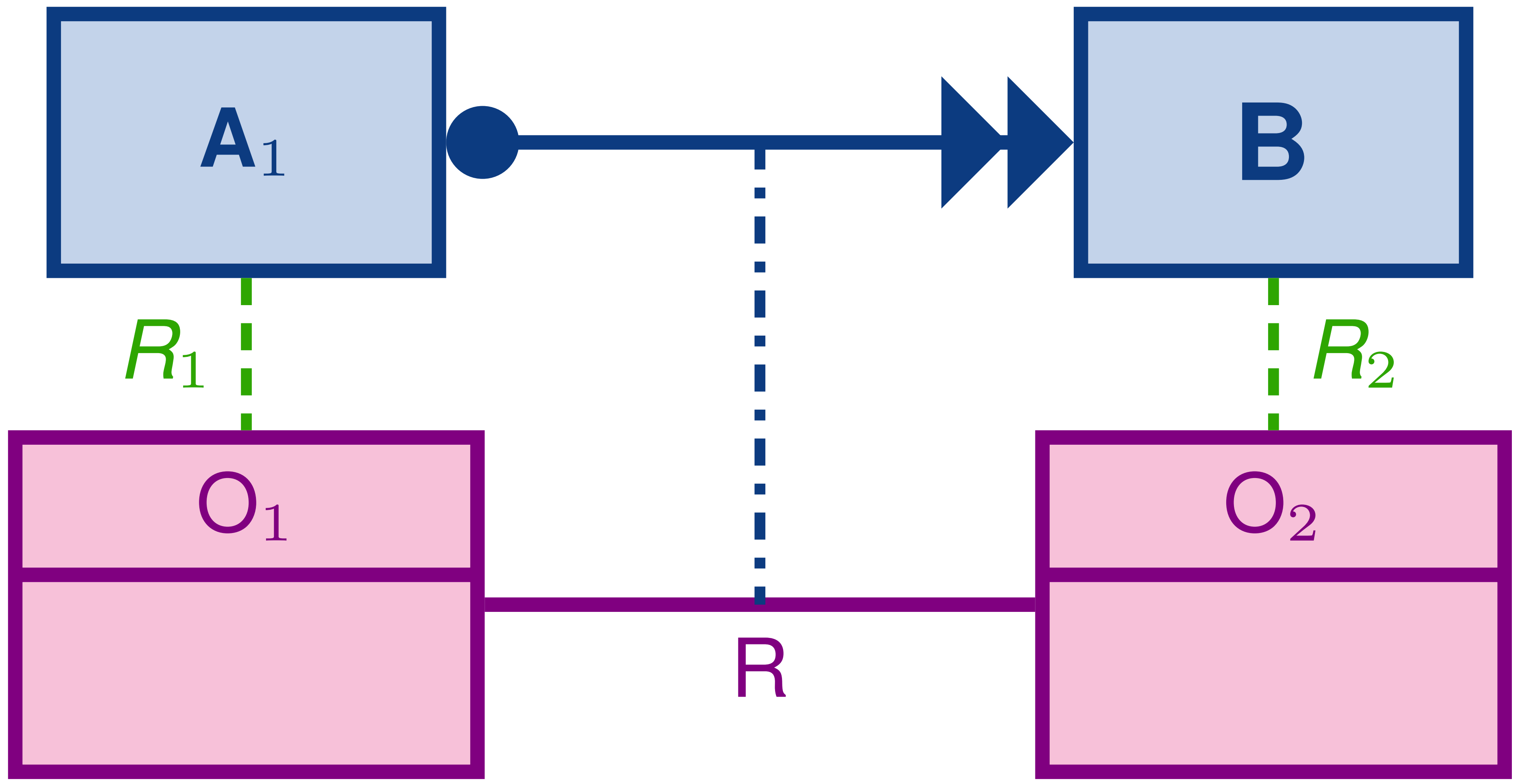


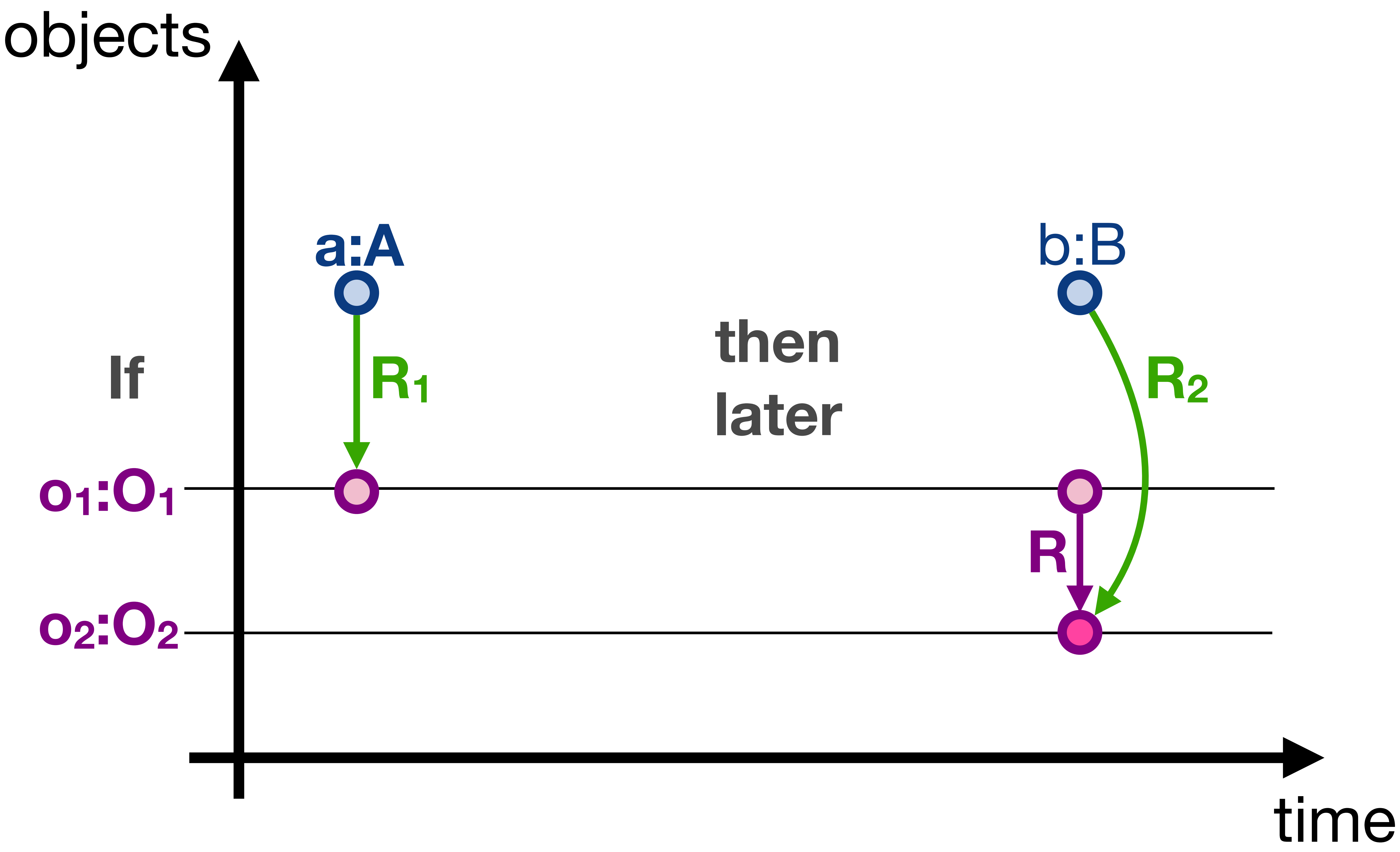
Object co-referencing on response





Relation co-referencing on response





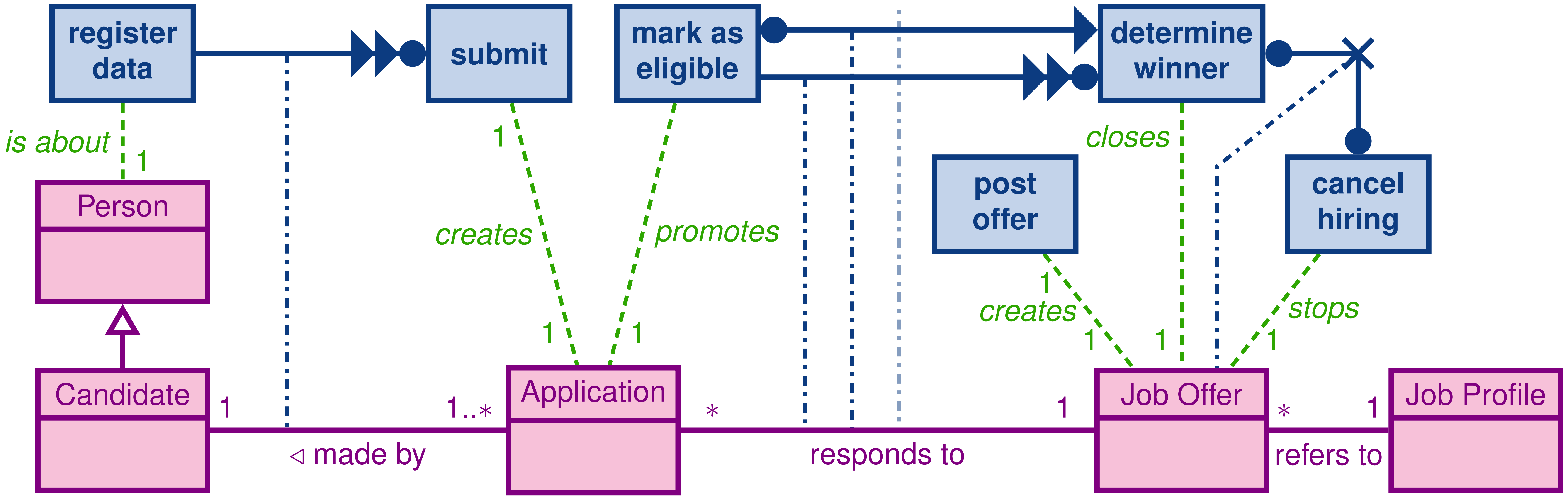
Object-centric behavioral constraints

Dimension 3: the process

- **constraints…**
- **…with data co-referencing**

A Job Offer *closed by* a **determine winner** task *cannot* be *stopped* by executing the **cancel hiring** task (and vice-versa).

An Application can be **submitted** only if, *beforehand*, the **data** *about* the Candidate who made *that* Application have been **registered**.

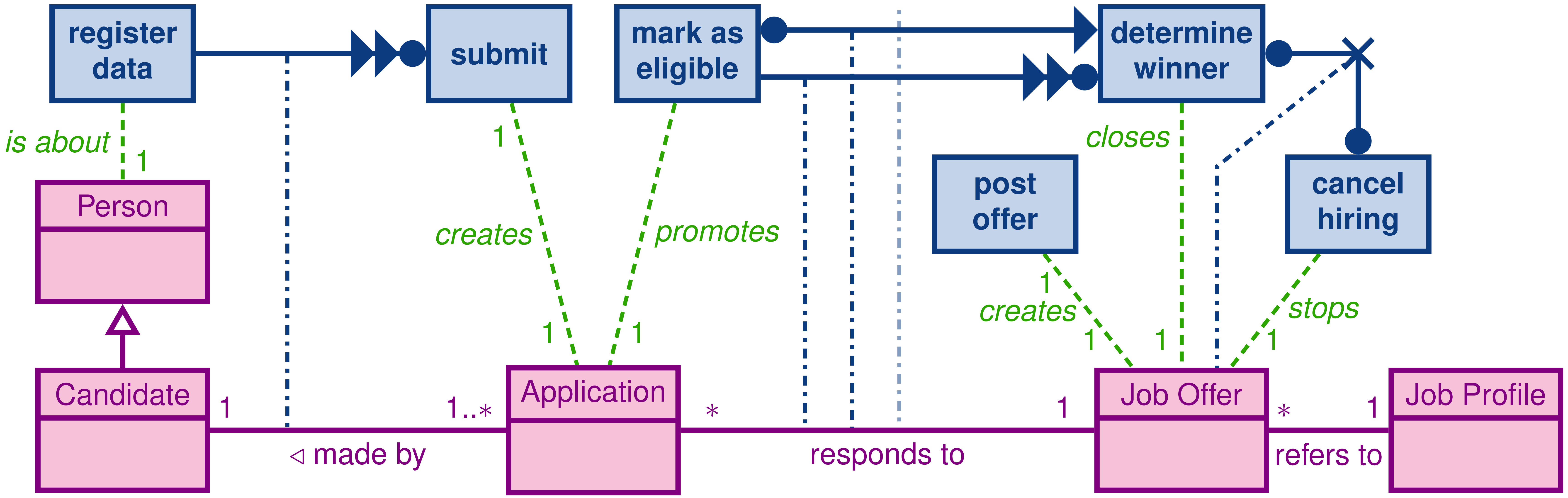


Object-centric behavioral constraints

Dimension 3: the process

- **constraints…**
- **…with data co-referencing**

A **winner** can be **determined** for a Job Offer only if *at least one* Application *responding to* *that* Job Offer has been *previously* **marked as eligible**. For each Application responding to a Job Offer, if the Application is **marked as eligible** then a **winner** must be *finally* **determined** for *that* Job Offer, and this is done *only once* for *that* Job Offer.

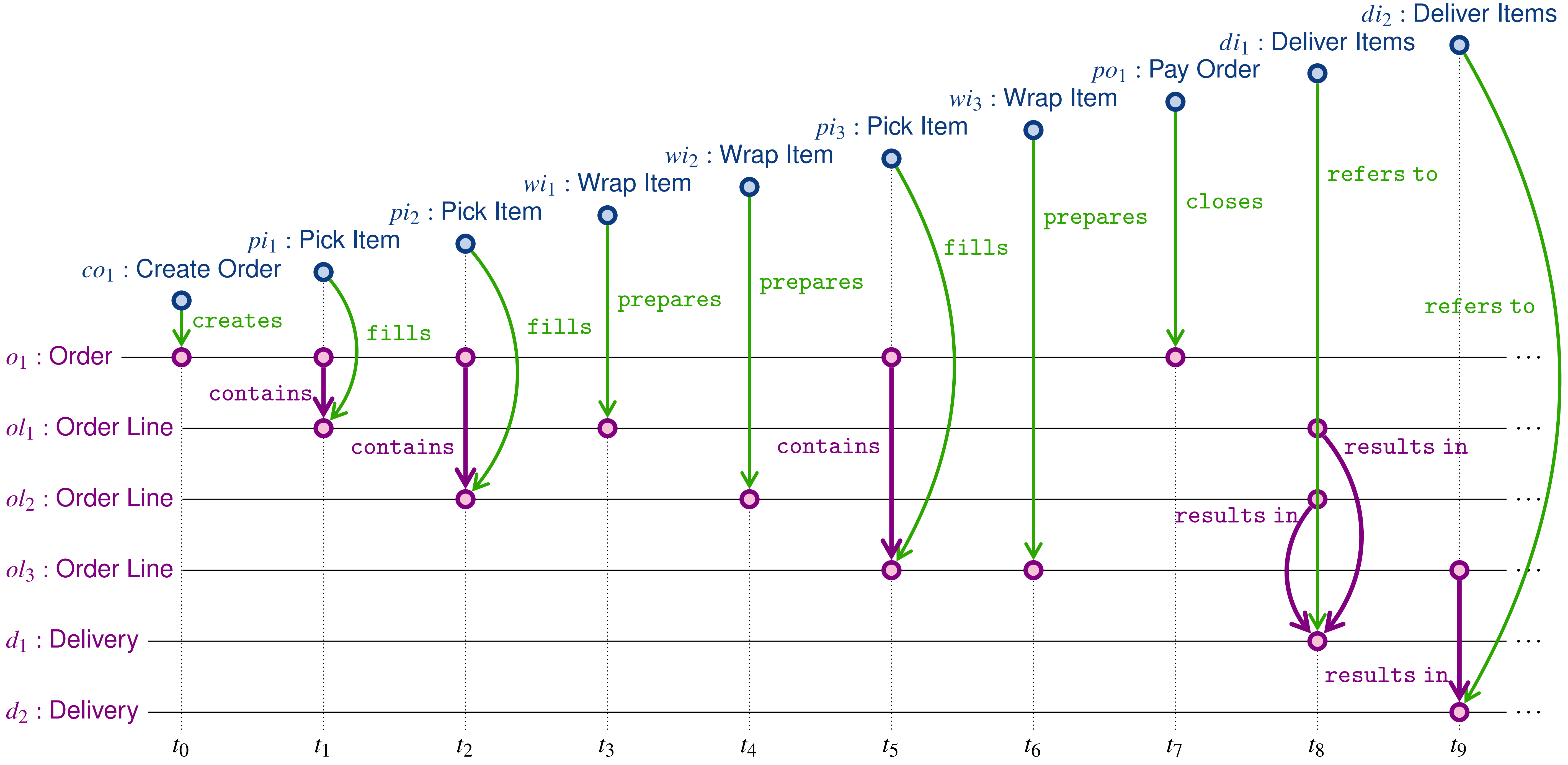


Semantics and formalization

Process execution: temporal knowledge graph

Data model: description logics

Object-centric constraints: temporal description logics



Achieved and ongoing results

Reasoning [____,BPM2019]

- Direct approach -> **undecidable**
- Careful **“object-centric” reformulation**
  -> **decidable** in EXPTIME (same as reasoning on UML diagrams)

Monitoring (ongoing)

- **Hybrid reasoning** (**closed** on the past, **open** on the future)

Discovery (ongoing)

- Construction of **trace views**
- **Standard discovery** on views
- Object-centric **reconciliation**

Conclusions

Augmented BPM: a framework for the intelligent management of processes at the intersection of **AI** and **BPM**

Central task: **framing**

Declarative approach: solid basis to framing with **uncertainty**, **data**, **objects and their interactions**

- **Reasoning** via **well-established formalisms and techniques**

Foundations well understood, effort needed towards **engineering**

Thanks to Wil van der Aalst, Anti Alman, Alessandro Artale, Federico Chesani, Giuseppe De Giacomo, Riccardo De Masellis, Claudio Di Ciccio, Marlon Dumas, Dirk Fahland, Paolo Felli, Alessandro Gianola, Alisa Kovtunova, Fabrizio Maggi, Andrea Marrella, Paola Mello, Jan Mendling, Fabio Patrizi, Rafael Penaloza, Maja Pesic, Andrey Rivkin, Michael Westergaard

Thank you!

montali@inf.unibz.it