Constraints for process framing in Augmented BPM

Marco Montali
Free University of Bozen-Bolzano

unibz

AI4BPM 2022, Münster, Germany

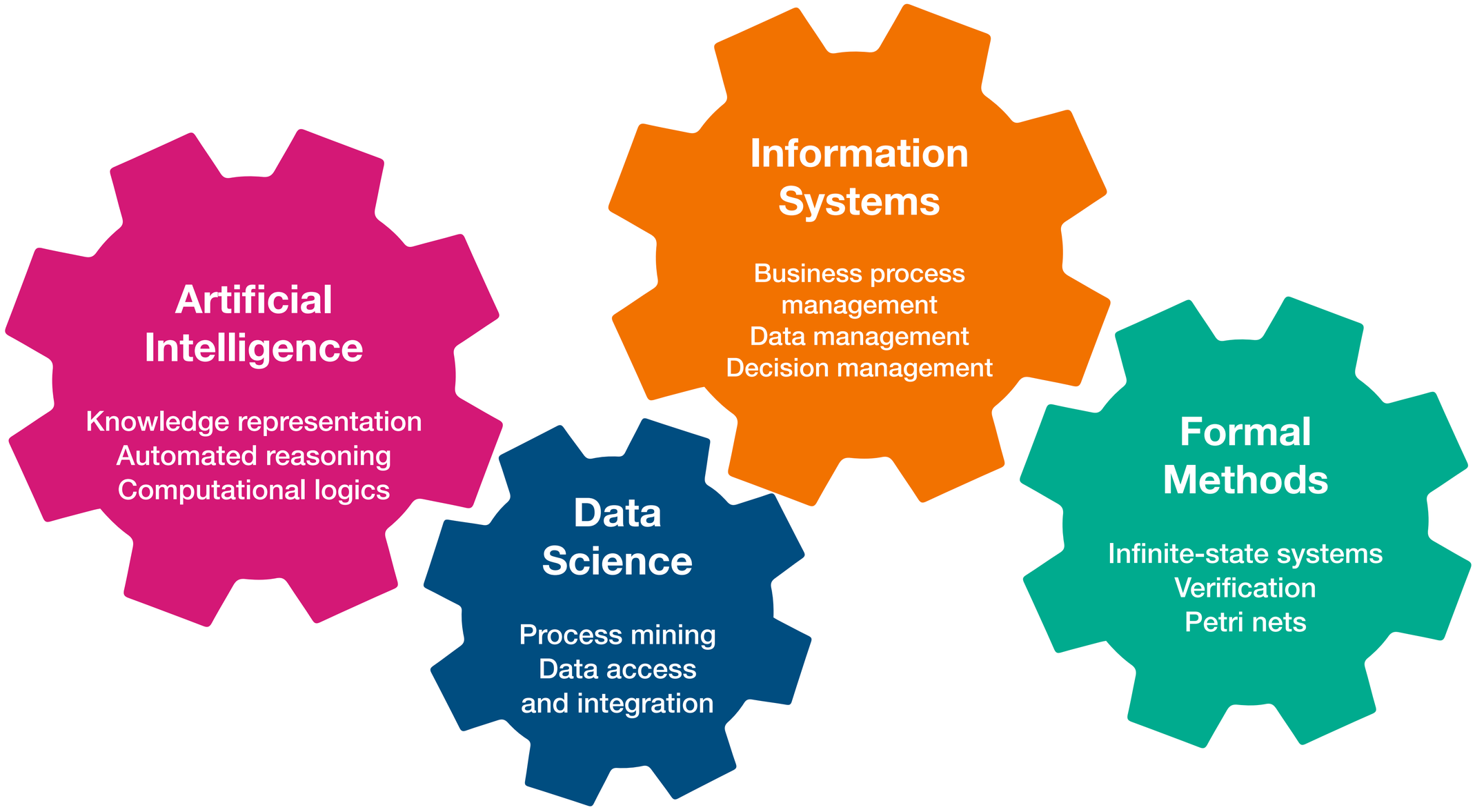
What do we do

We develop **foundational** and **applied techniques** grounded in **artificial intelligence**, **logics**, and **formal methods**,

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

processes and **data**.

How to attack these challenges?



Warm up:
what is augmented BPM

Augmented BPM

Computer Science > Artificial Intelligence

[Submitted on 30 Jan 2022 (v1), last revised 4 Aug 2022 (this version, v3)]

Augmented Business Process Management Systems: A Research Manifesto

Marlon Dumas, Fabiana Fournier, Lior Limonad, Andrea Marrella, Marco Montali, Jana-Rebecca Rehse, Rafael Accorsi, Diego Calvanese, Giuseppe De Giacomo, Dirk Fahland, Avigdor Gal, Marcello La Rosa, Hagen Völzer, Ingo Weber

Augmented BPM

An increased availability of **business process execution data**, combined with **advances in AI**, have laid the ground for the emergence of **information systems** where the **execution flows are not pre-determined**, **adaptations do not require explicit changes to software** applications, and **improvement opportunities are autonomously discovered**, **validated**, and **enabled on-the-fly**.

Augmented Business Process Management Systems: A Research Manifesto

Marlon Dumas, Fabiana Fournier, Lior Limonad, Andrea Marrella, Marco Montali, Jana-Rebecca Rehse, Rafael Accorsi, Diego Calvanese, Giuseppe De Giacomo, Dirk Fahland, Avigdor Gal, Marcello La Rosa, Hagen Völzer, Ingo Weber

Augmented BPM System

AI-empowered, **trustworthy**, and **process-aware information system** that
reasons and acts upon data within a **set of constraints** and **assumptions**
with the aim to
continuously adapt and improve a **set of business processes** with respect to one or more **performance indicators**

Augmented BPM System

AI-empow ss-aware
informatio
reasons ar t of
constraints
with the aim
continuousl f business
processes w formance
indicators

Strongly related to
BPM and integrative AI

Thu 09:00-10:00
Keynote by
Chiara Ghidini

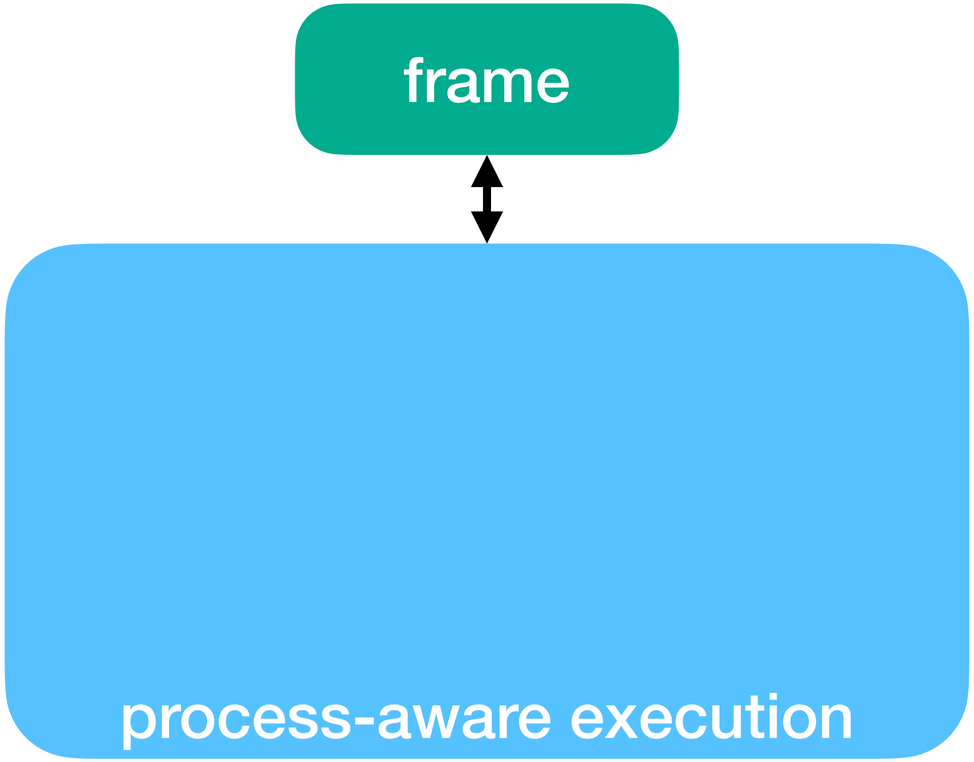
Data, Conceptual Knowledge, and AI: What can they do together?

ABPM lifecycle

Revisiting the BPM lifecycle: from “design” to “framing”

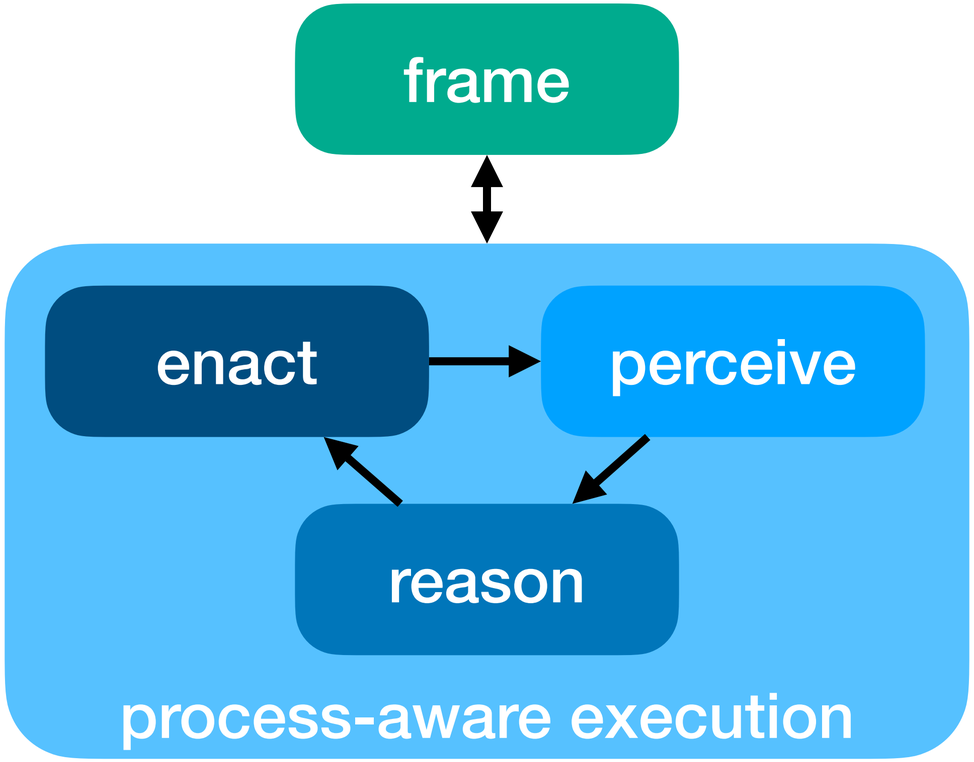
ABPM lifecycle

Revisiting the BPM lifecycle: continuous evolution



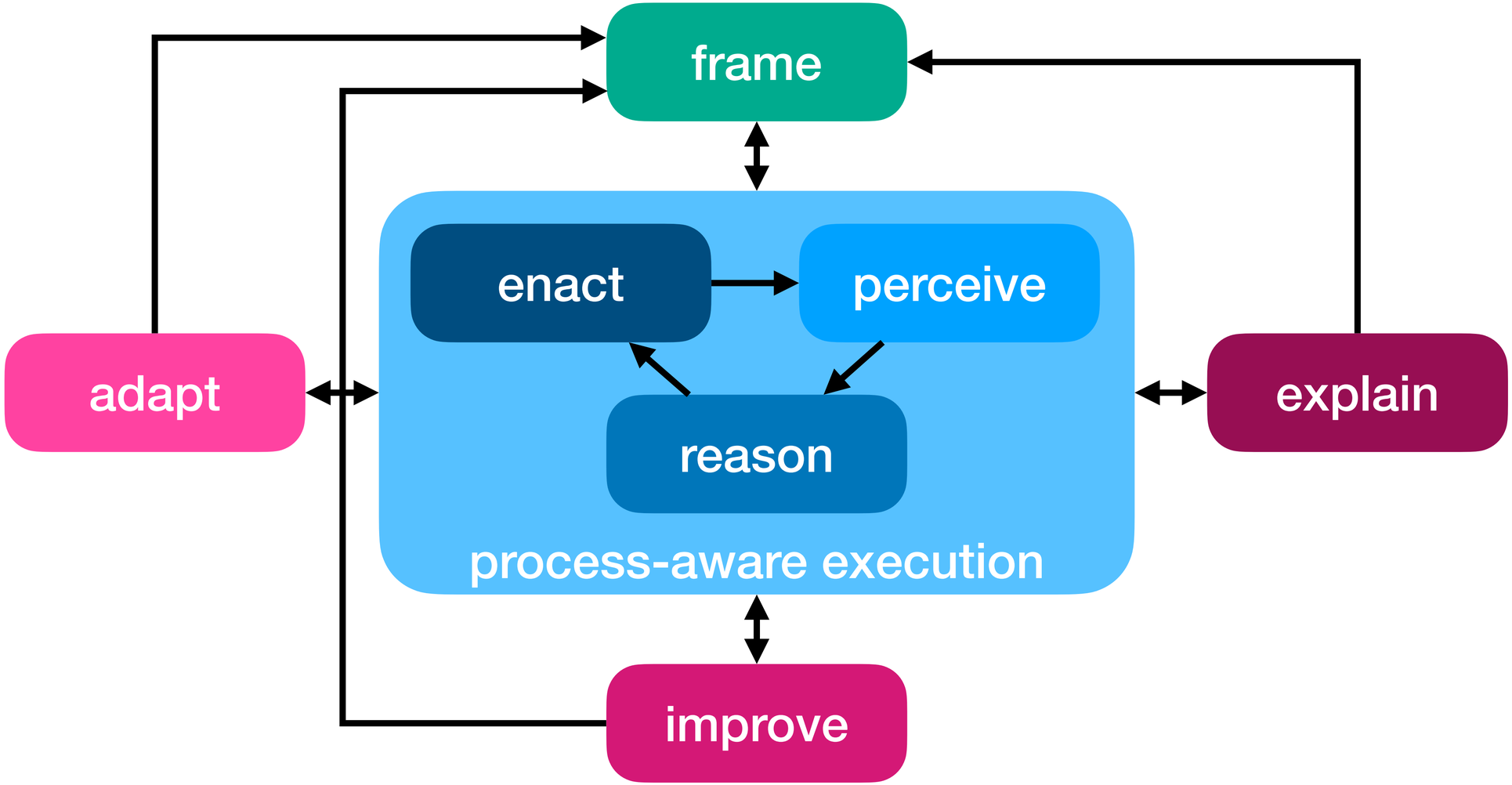
ABPM lifecycle

Revisiting the BPM lifecycle: continuous evolution



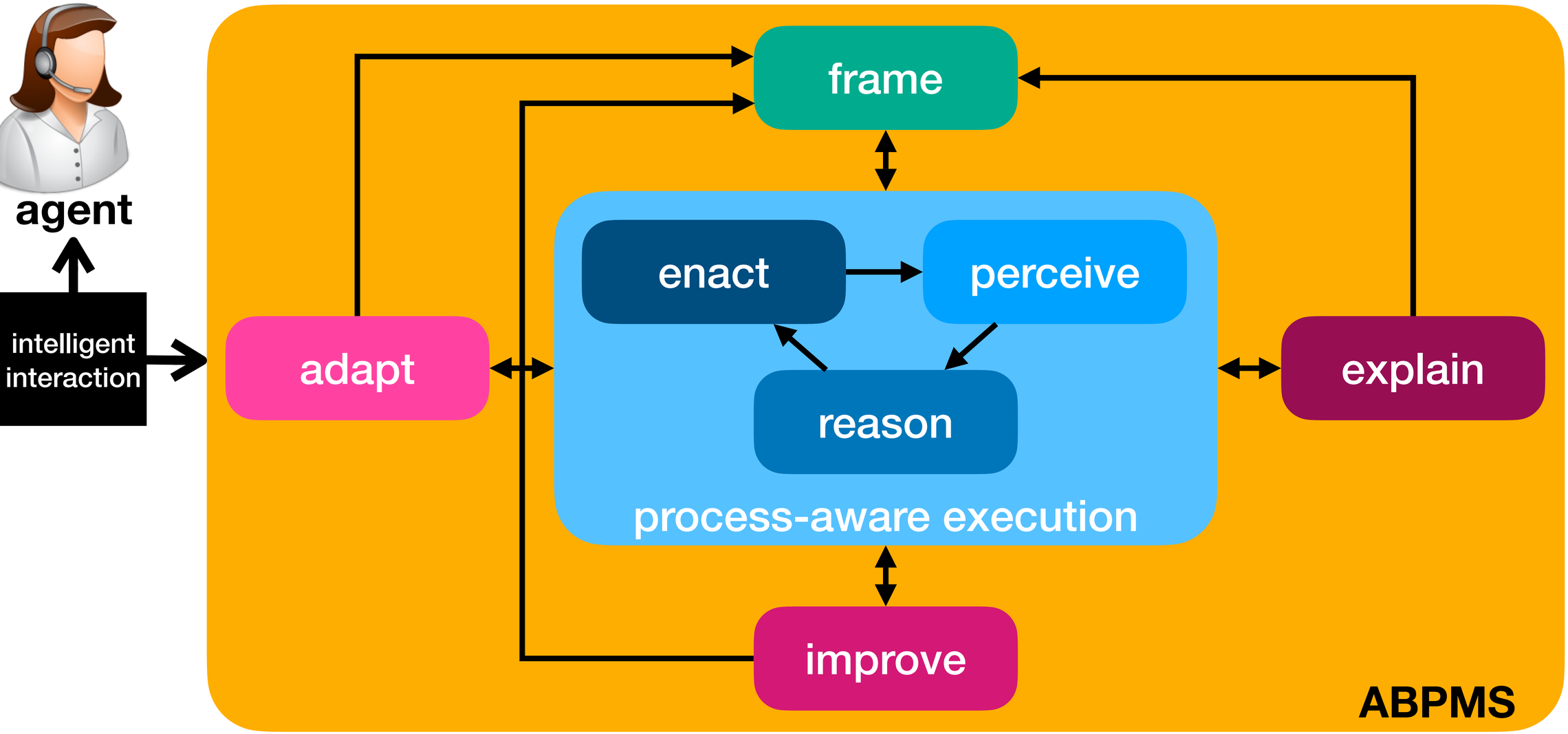
ABPM lifecycle

Extending with “pure” AI capabilities



ABPM lifecycle

Continuous interaction with principals



Features of an ABPMS

Framed autonomy

ABPMS acts **autonomously**

- Lifecycle steps performed proactively and continuously

ABPMS acts **“within its frame”**

- Maximally permissive, goal-driven strategy

Features of an ABPMS

Framed autonomy

ABPMS acts **autonomously**

- Lifecycle steps performed proactively and continuously

ABPMS acts **“within its frame”**

- Maximally permissive, goal-driven strategy

What does this mean?

Hard vs soft constraints, reframing, meta-framing, …

Features of an ABPMS

Conversationally actionable

Autonomy does not mean isolation

- Need of **continuous conversation** with human principal(s)

Conversational

- Language-based communication with humans (proactive and reactive)

Actionable

- Interaction leads to actual decision making

Features of an ABPMS

Conversationally actionable

Autonomy does not mean isolation

- Need of **continuous conversation** with human principal(s)

Conversational

- Language-bas humans (proactive and reactive)

Which focus?

Actions, goals, intensions, but also models!

Actionable

- Interaction leads to actual decision making

Features of an ABPMS

Adaptive

Motto: **react to changes** and **self-improve**

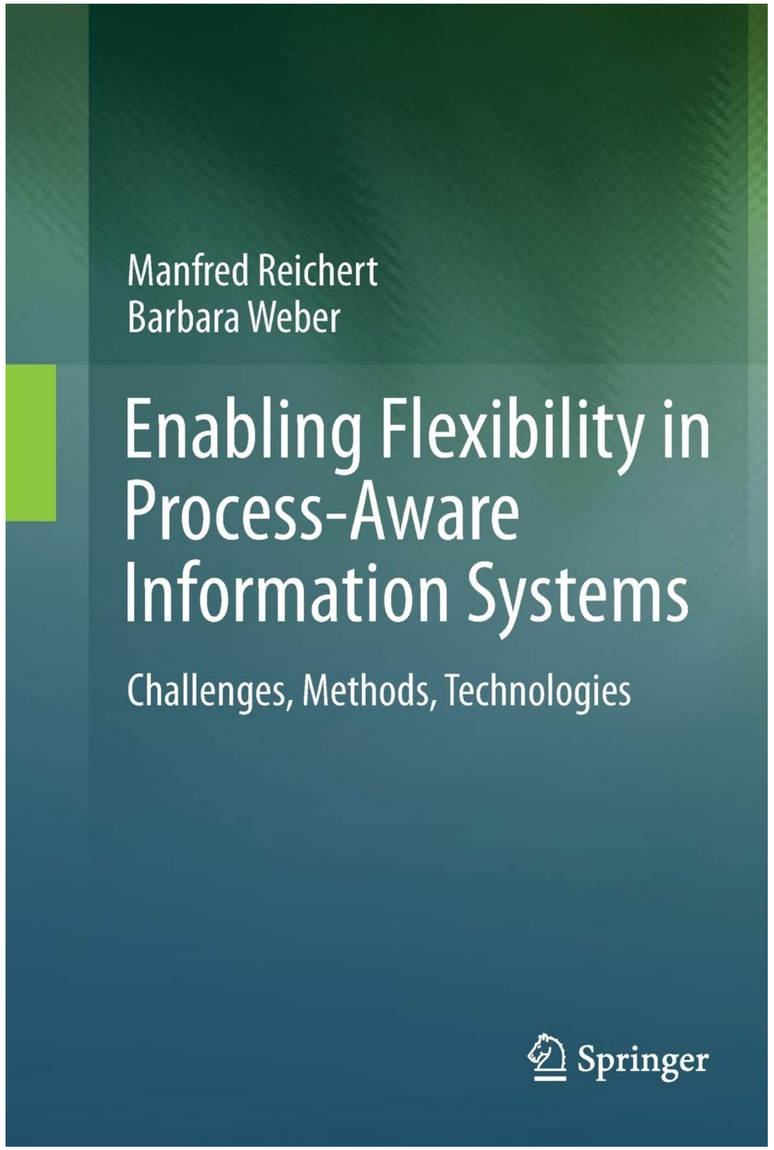
- Prediction
- Instance- and model-level adaptations

Features of an ABPMS

Adaptive

Motto: **react to changes** and **self-improve**

- Prediction
- Instance- and model-level adaptations



Features of an ABPMS

Adaptive

Motto: **react to changes** and **self-improve**

- Prediction
- Instance- and model-level adaptations
- **Multi-objective optimisation**
- Evaluation of **trade-offs**

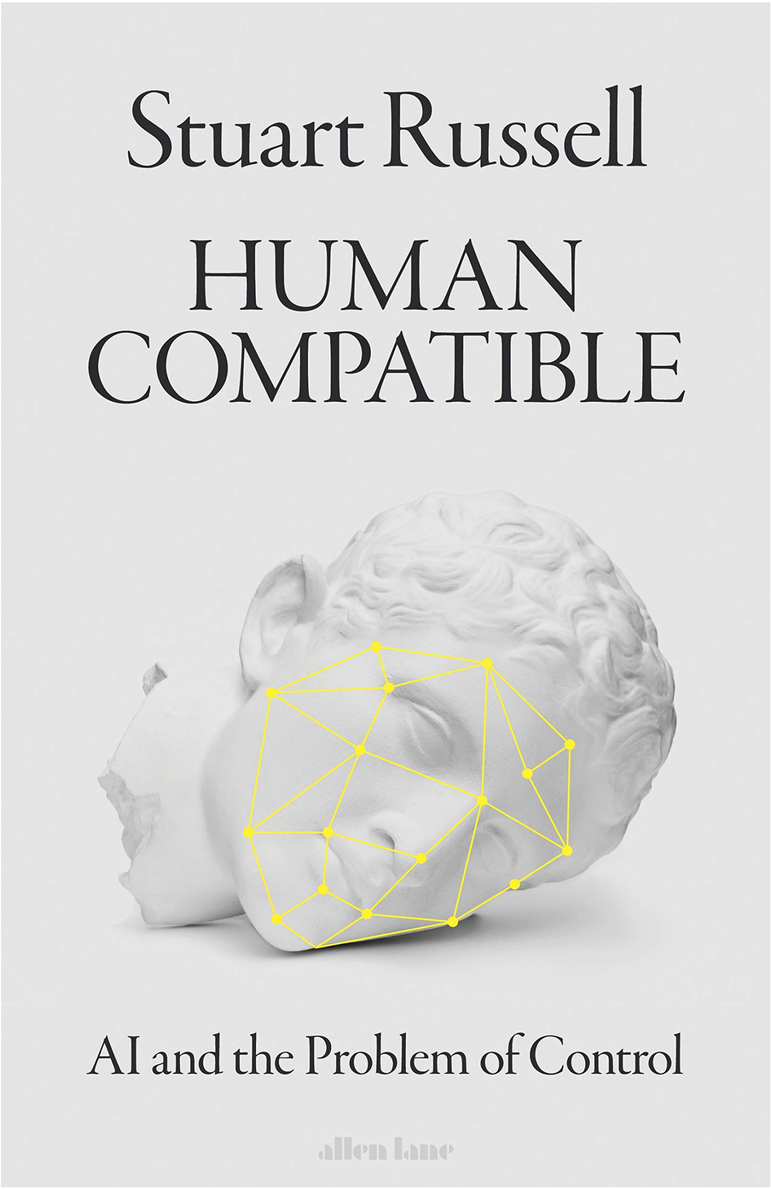
Features of an ABPMS

Adaptive

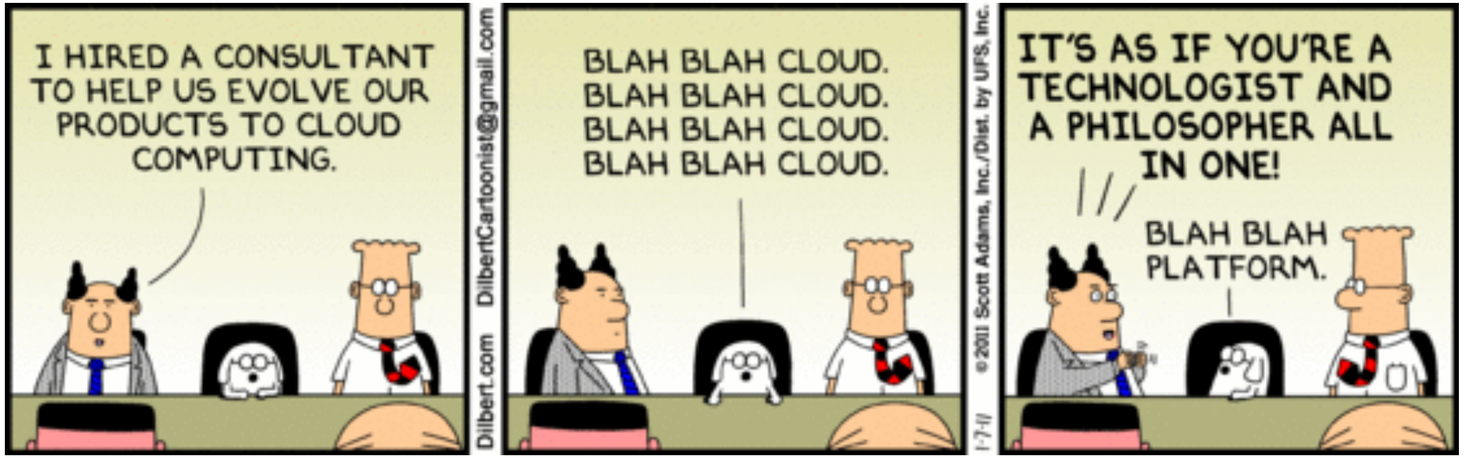
Motto: **react to changes** and **self-improve**

- Prediction
- Instance- and model-level adaptations
- **Multi-objective optimisation**
- Evaluation of **trade-offs**

What if there are multiple principals?



Quiz: which feature is missing?



Features of an ABPMS

Explainable

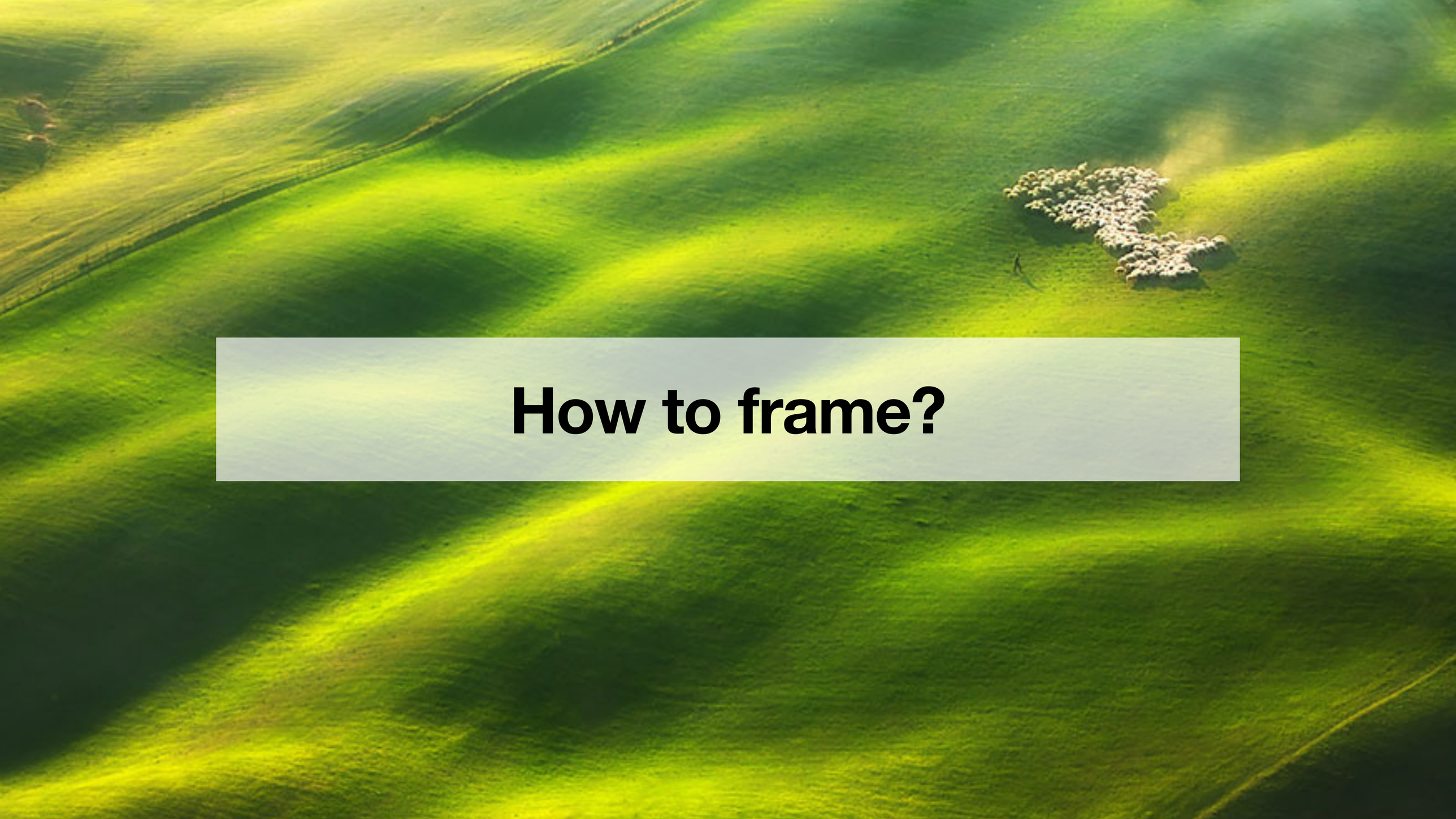
An ABPMS should be **trustworthy**

- *“trust is the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party”* **[MayerEtAl,TAMR95]**

How to be trustworthy?

- **Fair**
- **Explainable**
- **Auditable**
- **Safe**

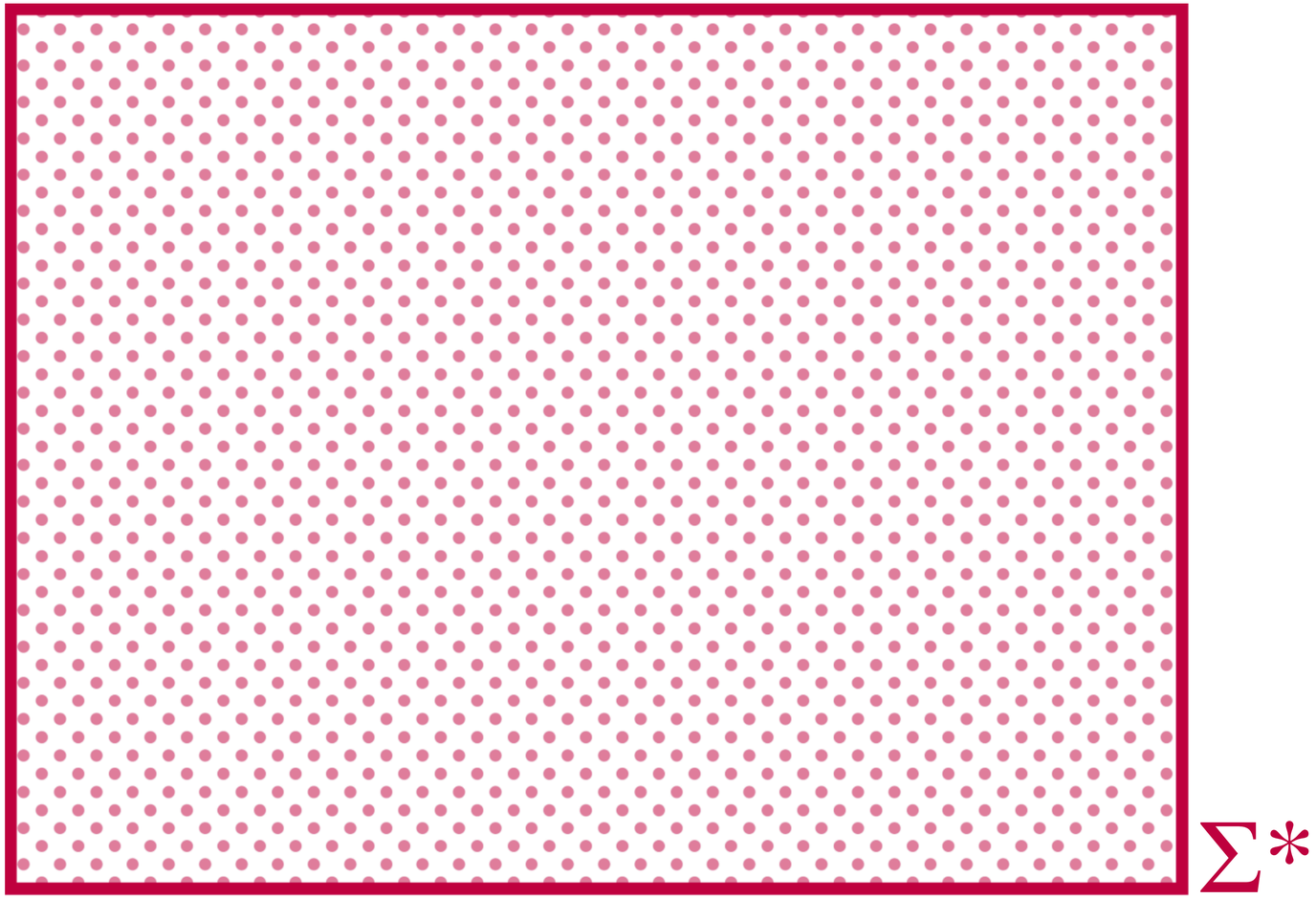
Need of specific focus on “process-aware” systems



How to frame?

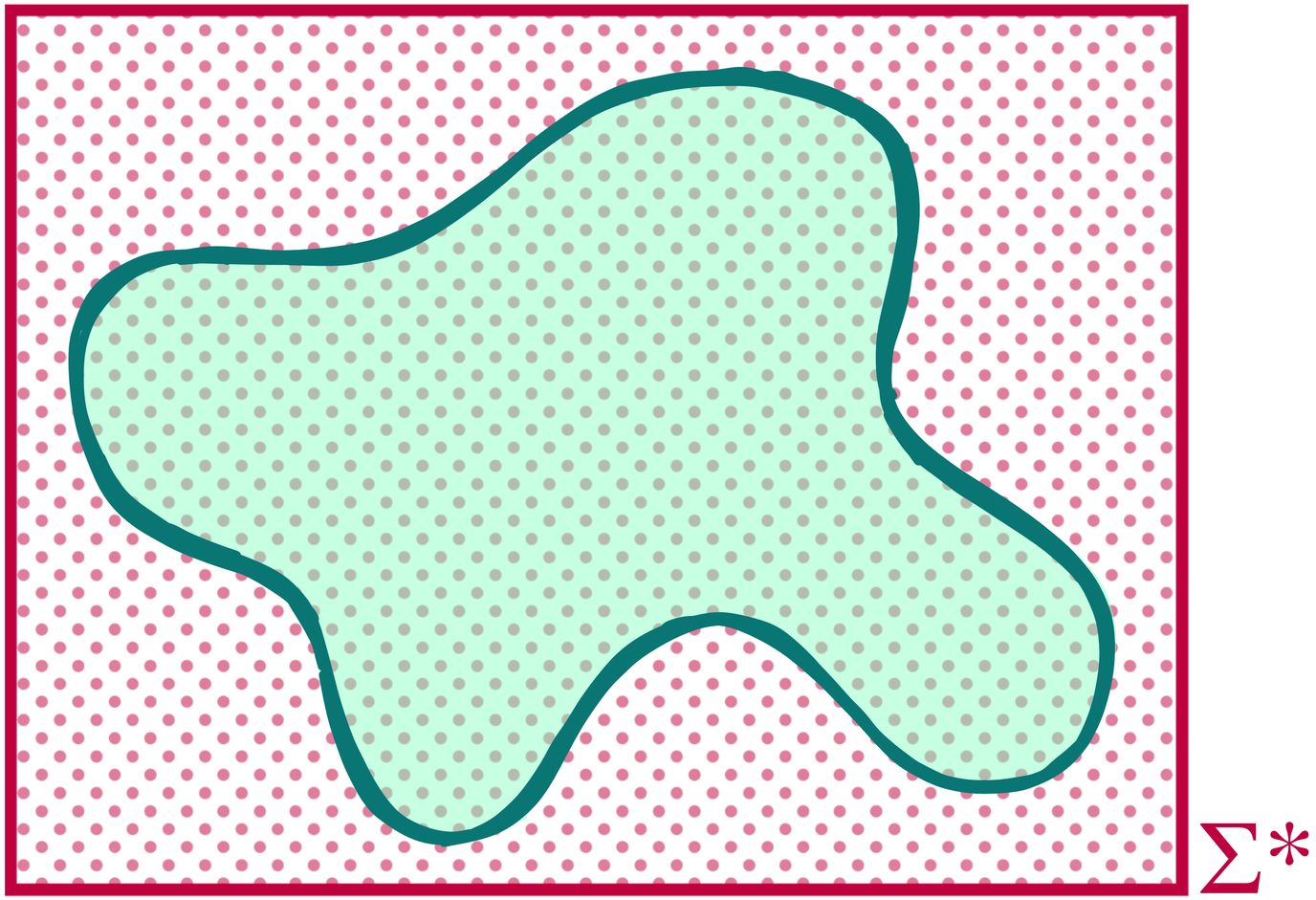
What is a process?

A possibly infinite set of finite traces

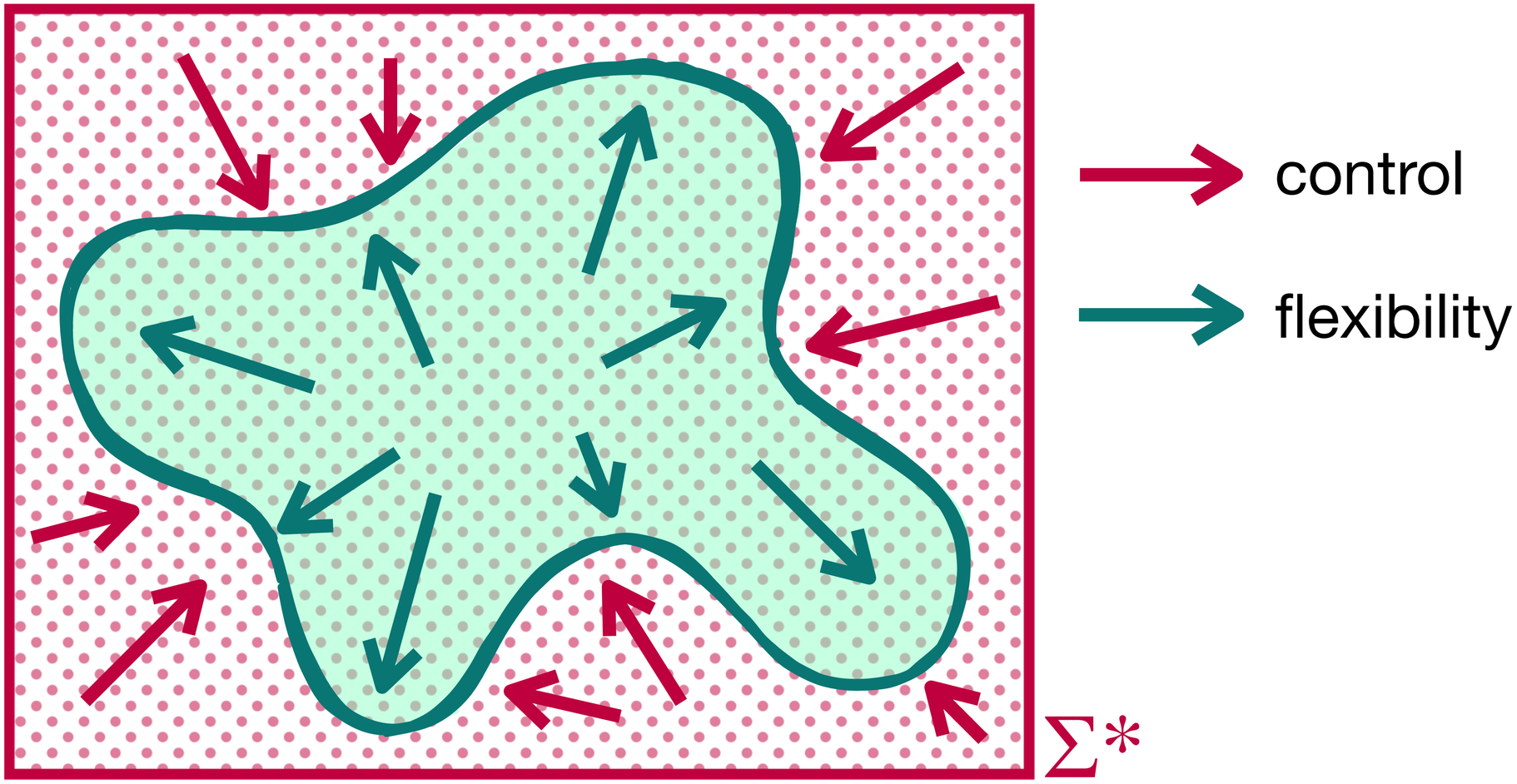


What is a process?

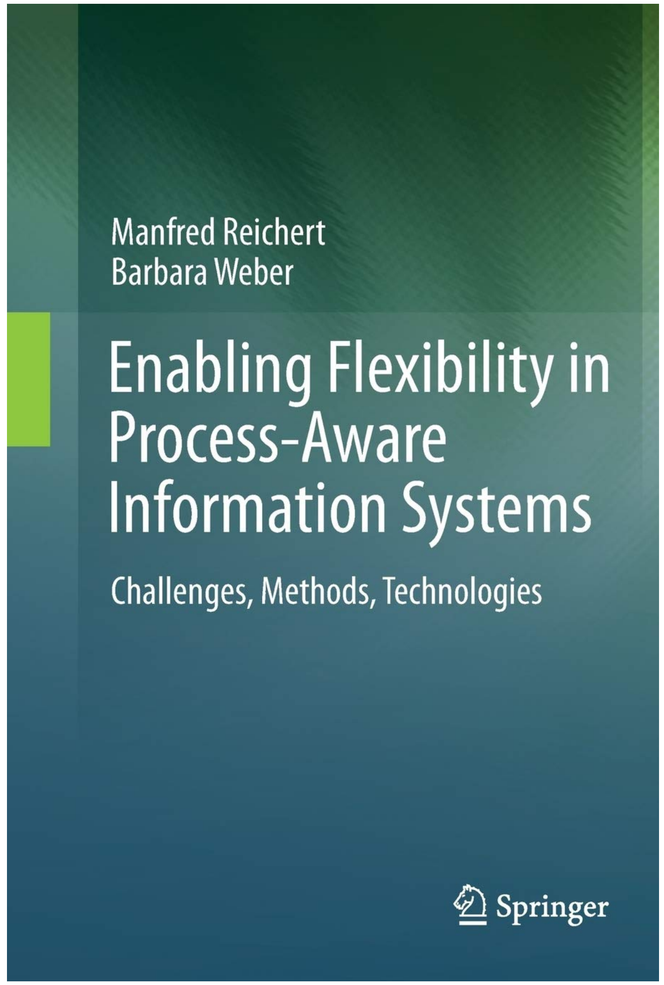
A possibly infinite set of finite traces



Flexibility and control as contrasting forces



The issue of flexibility is widely known

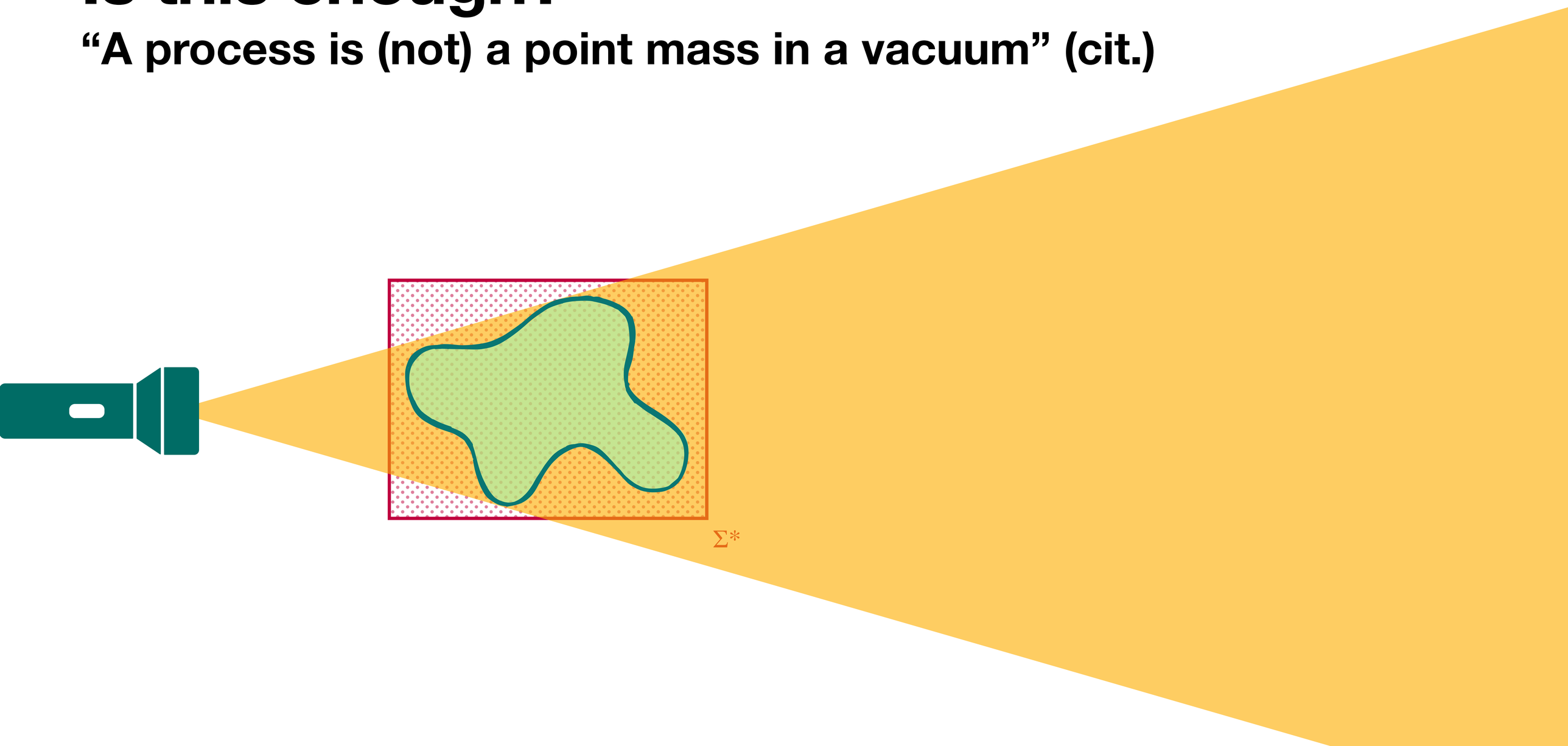


Different ways to address flexibility in processes

We are interested here in **flexibility by design**

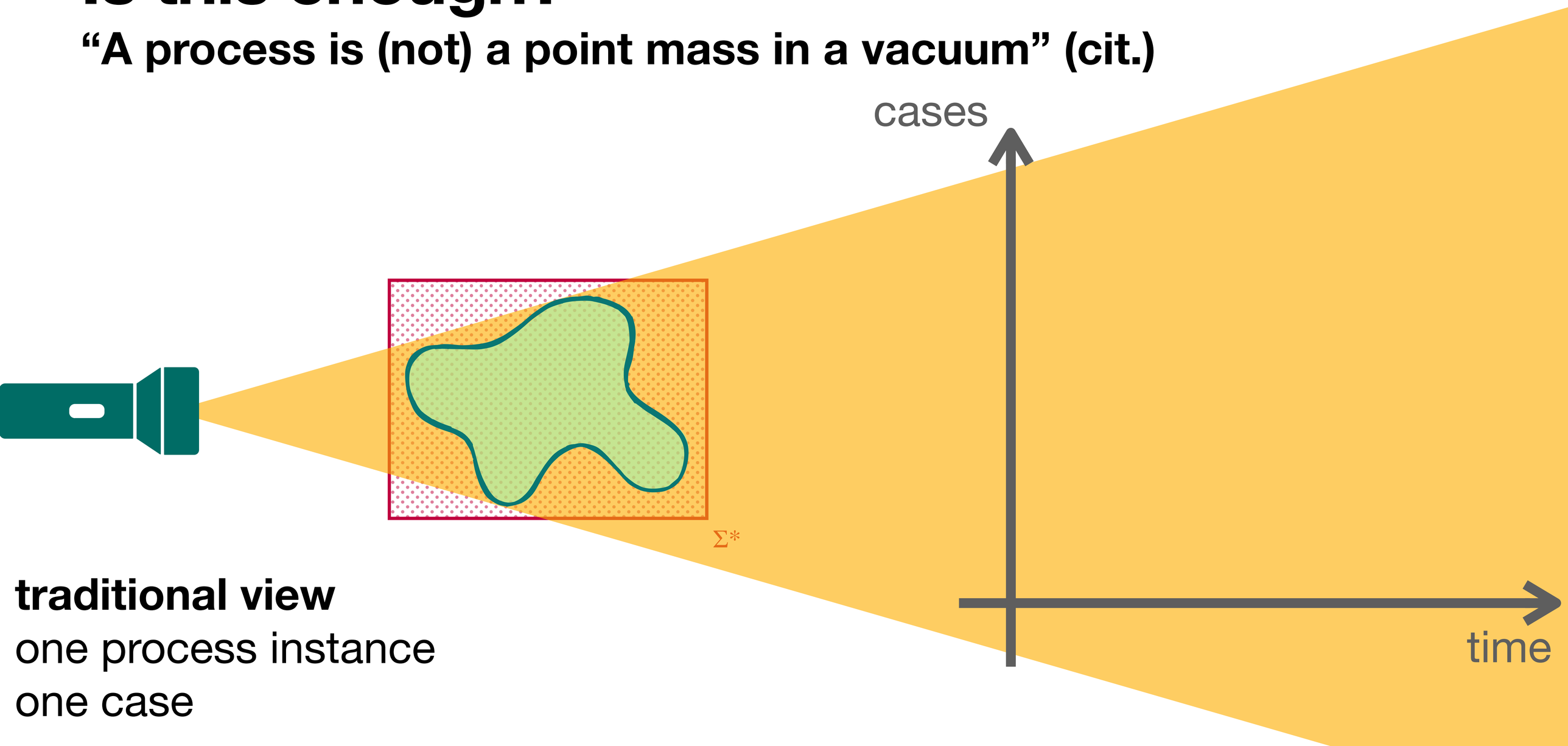
Is this enough?

“A process is (not) a point mass in a vacuum” (cit.)



Is this enough?

“A process is (not) a point mass in a vacuum” (cit.)



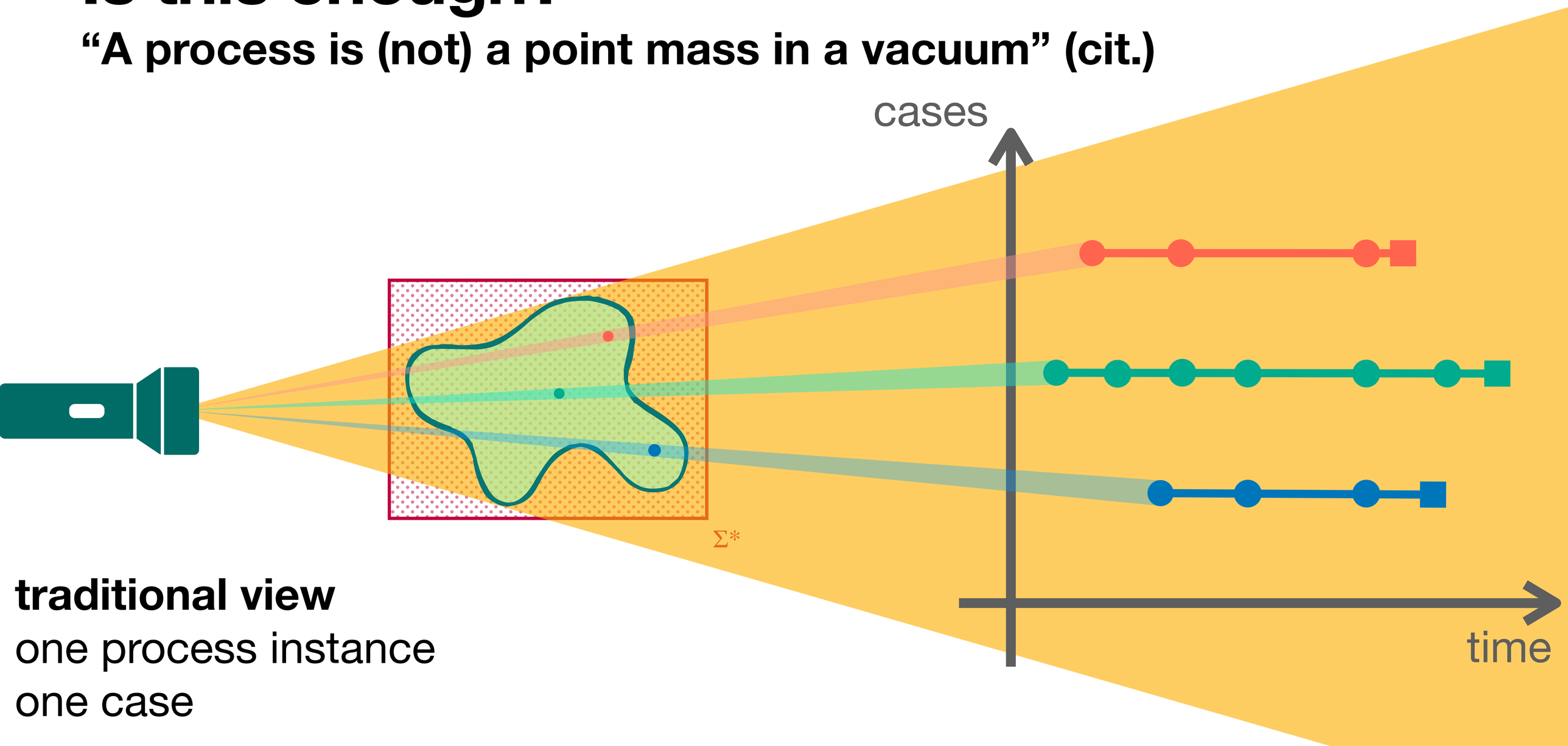
traditional view

one process instance

one case

Is this enough?

“A process is (not) a point mass in a vacuum” (cit.)



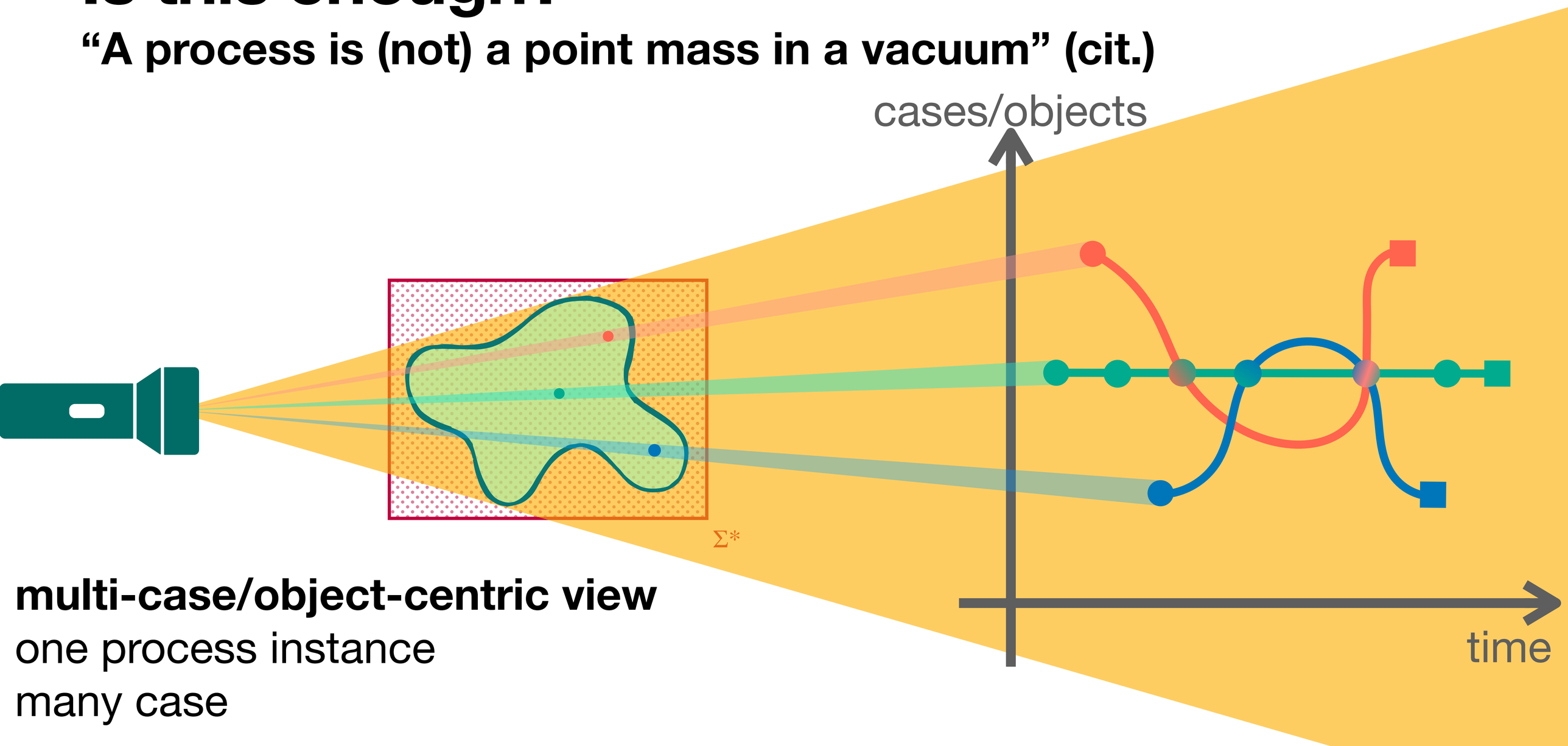
traditional view

one process instance

one case

Is this enough?

“A process is (not) a point mass in a vacuum” (cit.)



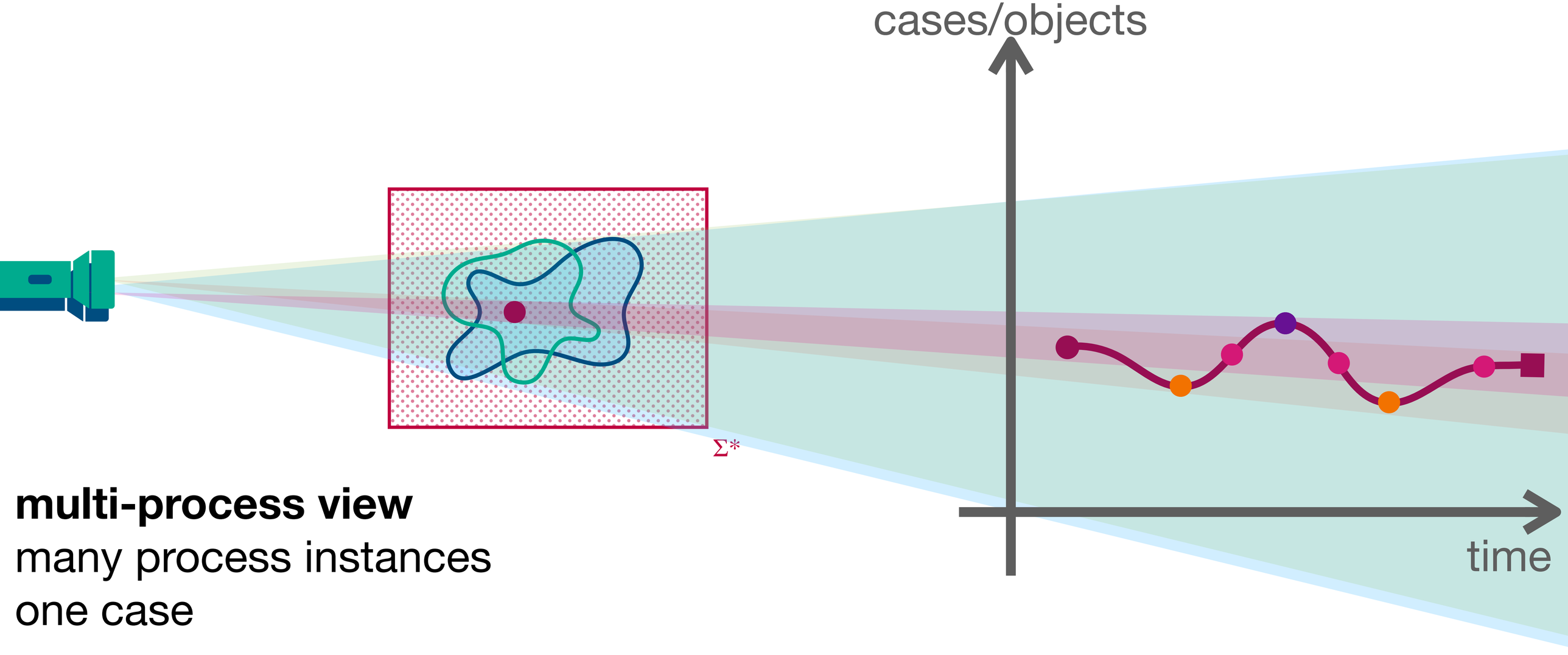
multi-case/object-centric view

one process instance

many case

Is this enough?

“A process is (not) a point mass in a vacuum” (cit.)

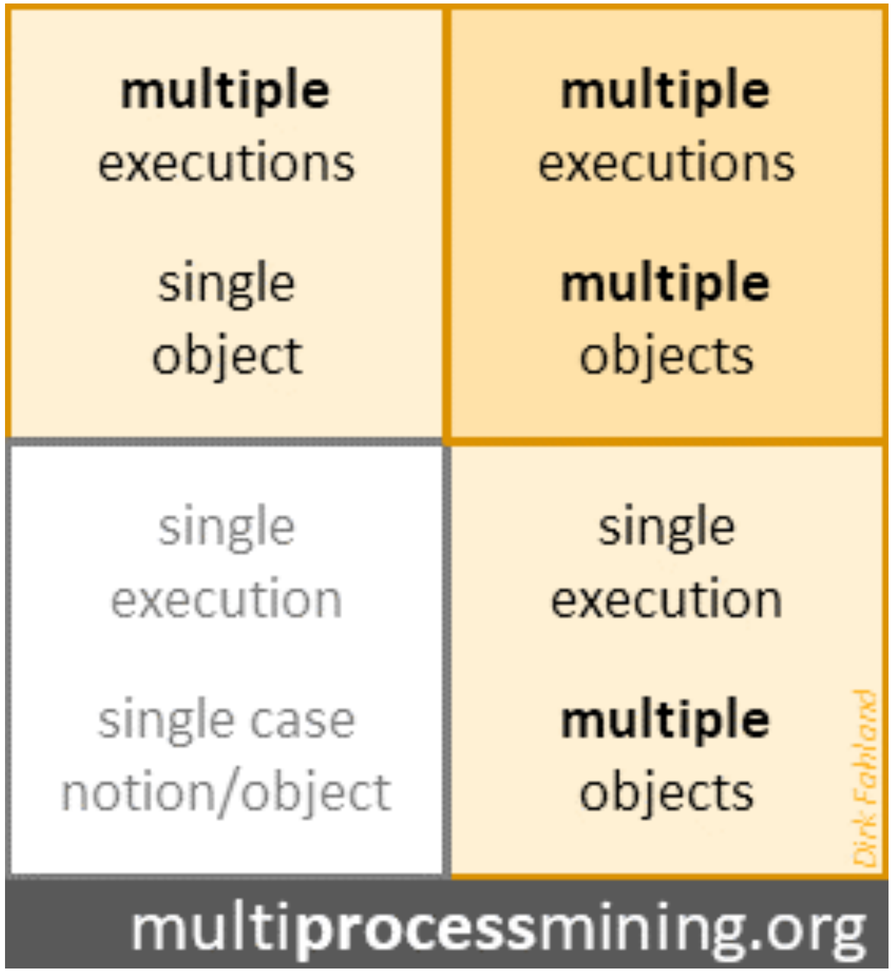


multi-process view

many process instances

one case

More in general…

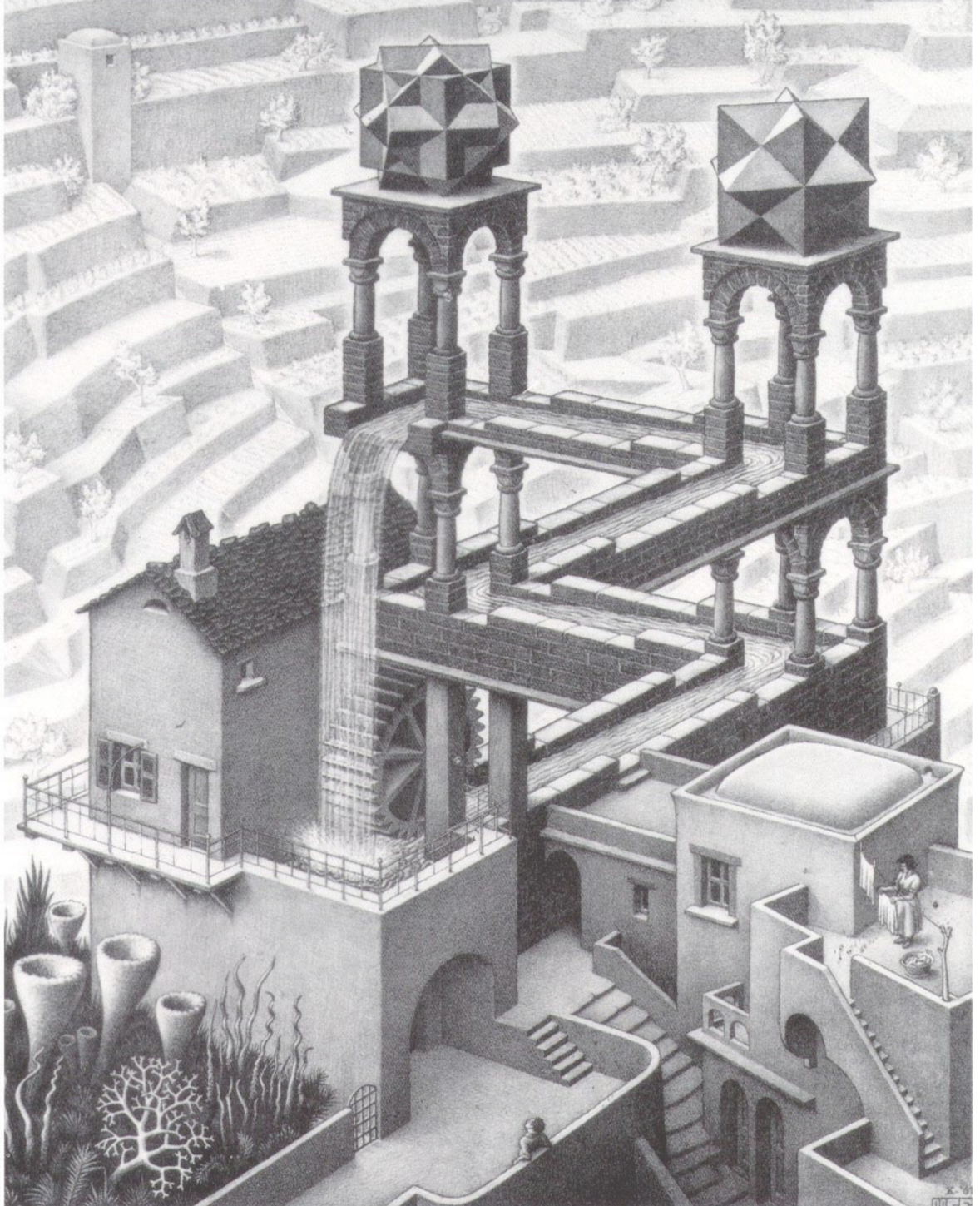


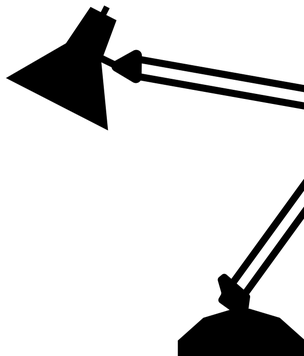
Wed 10:30-12:30

Tutorial by
Dirk Fahland

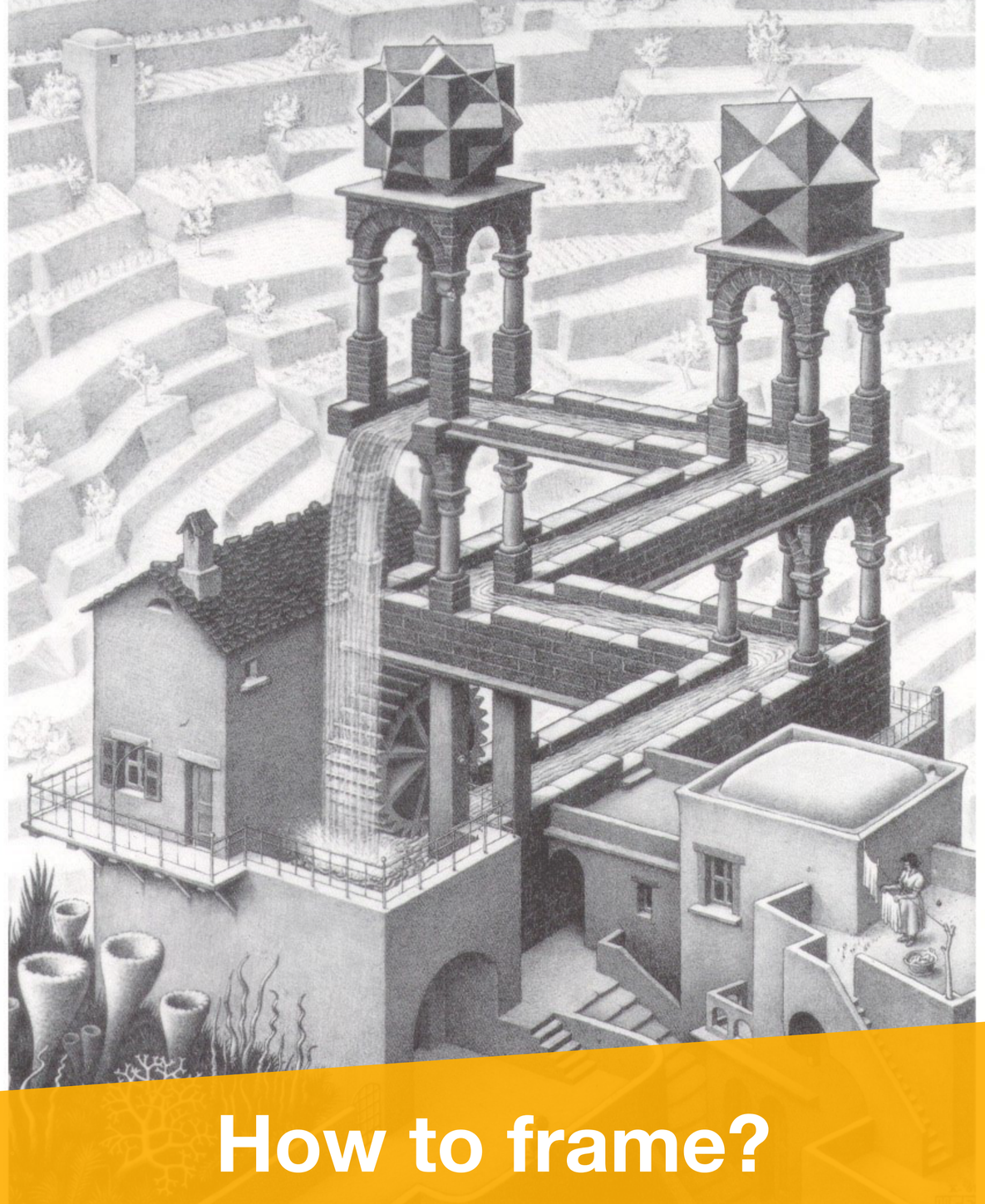
Multi-dimensional process analysis

Outline



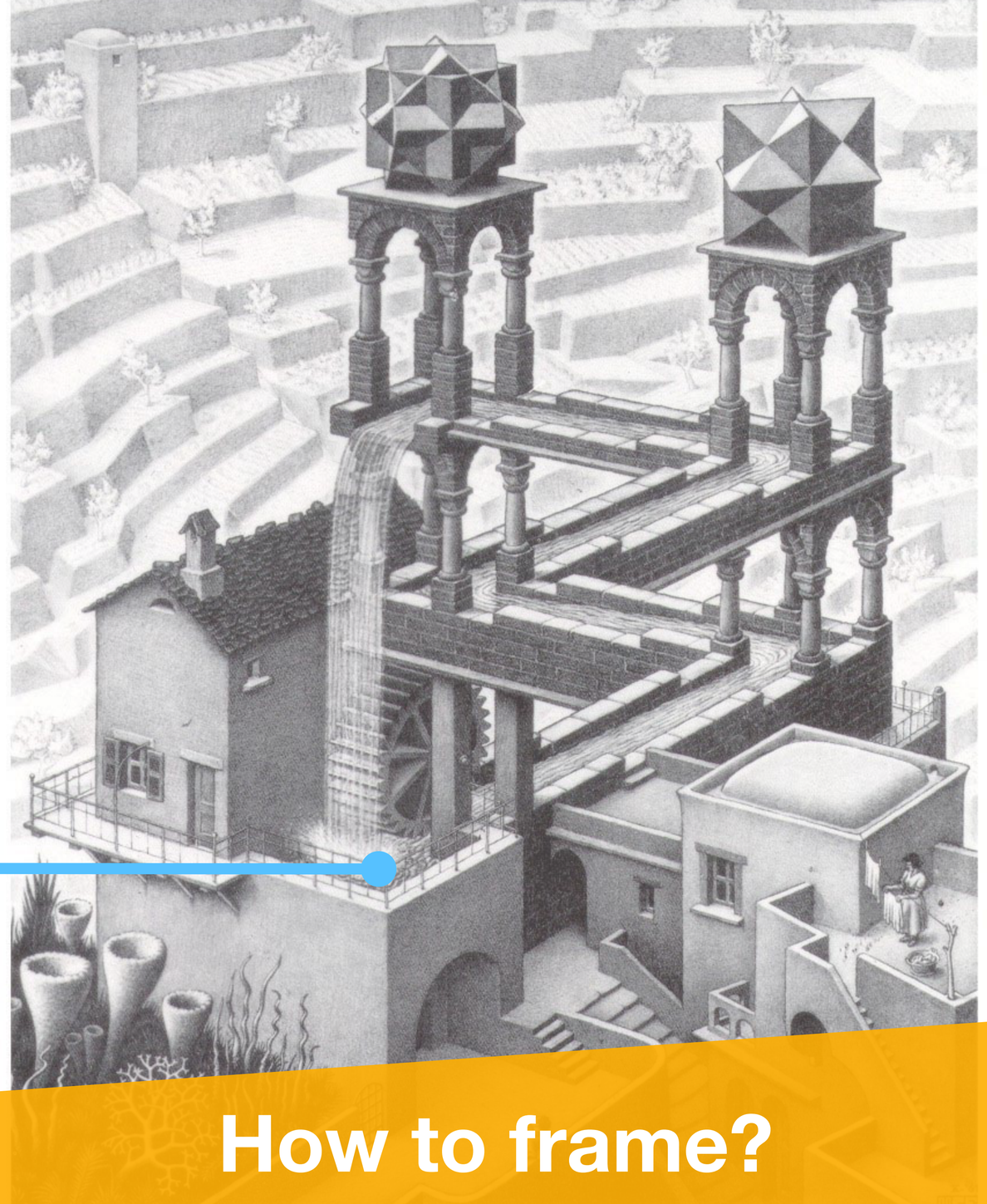


Outline



How to frame?

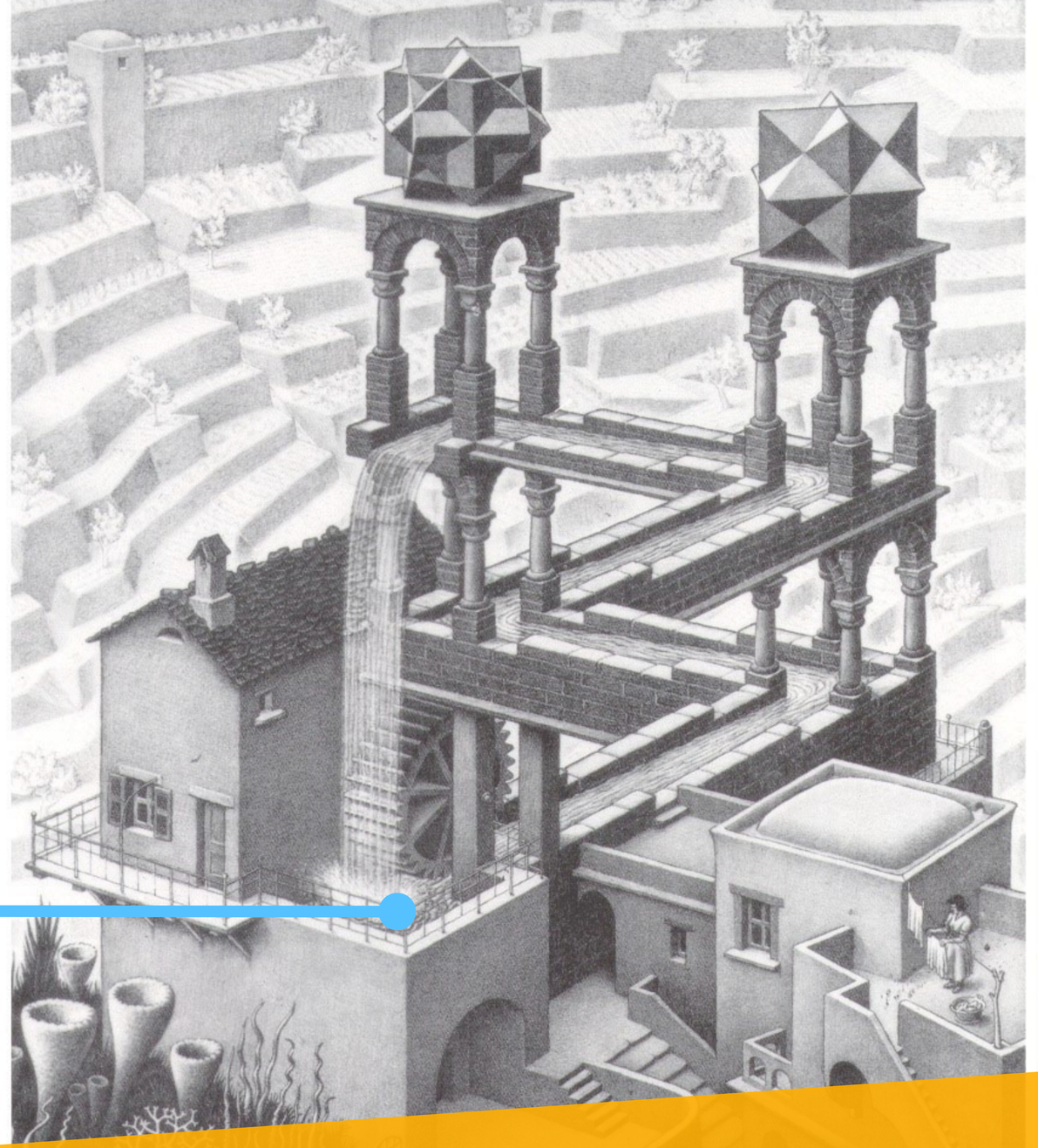
Outline



the declarative approach

How to frame?

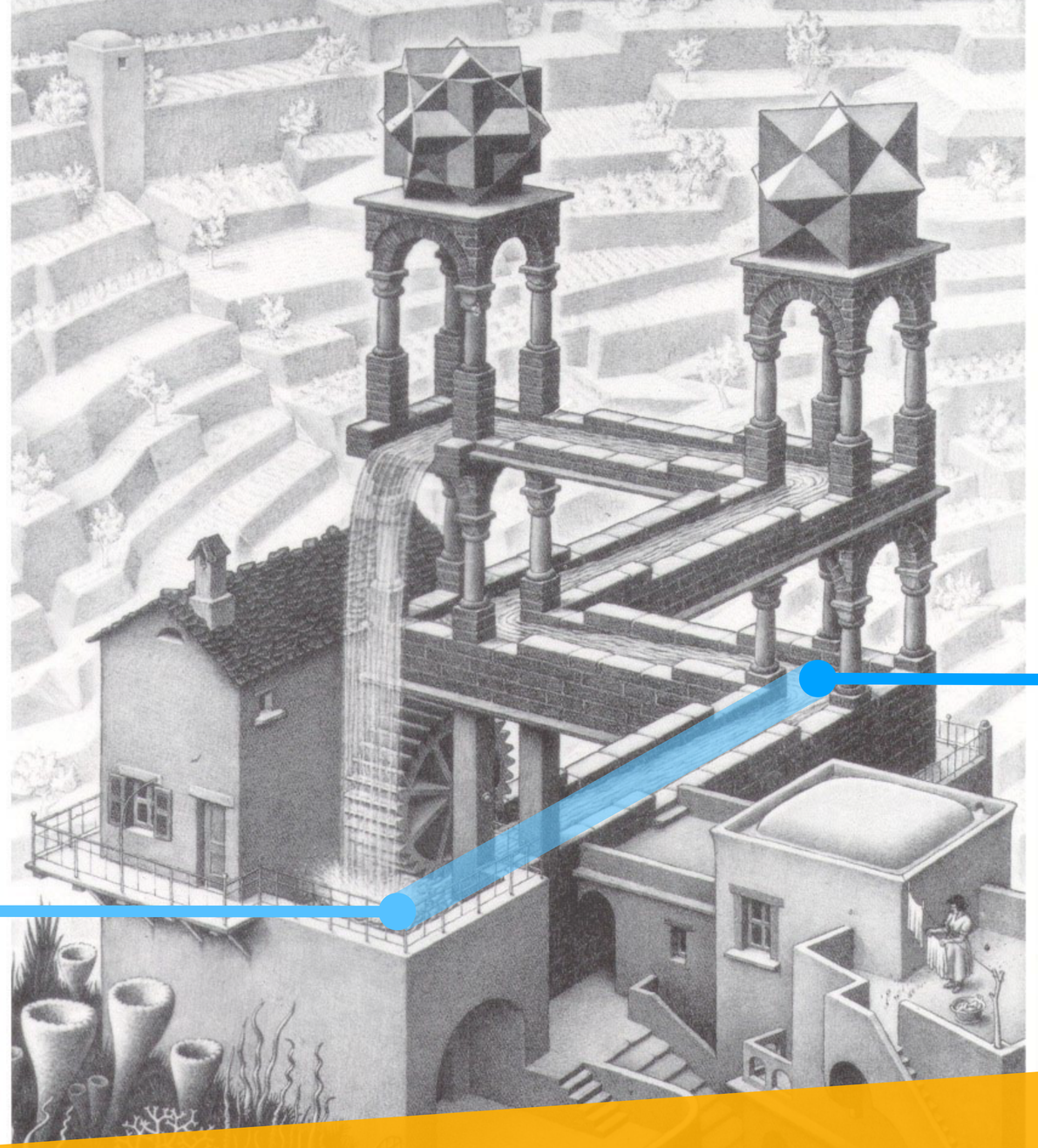
Outline



the declarative approach

How to deal with deviations?

Outline

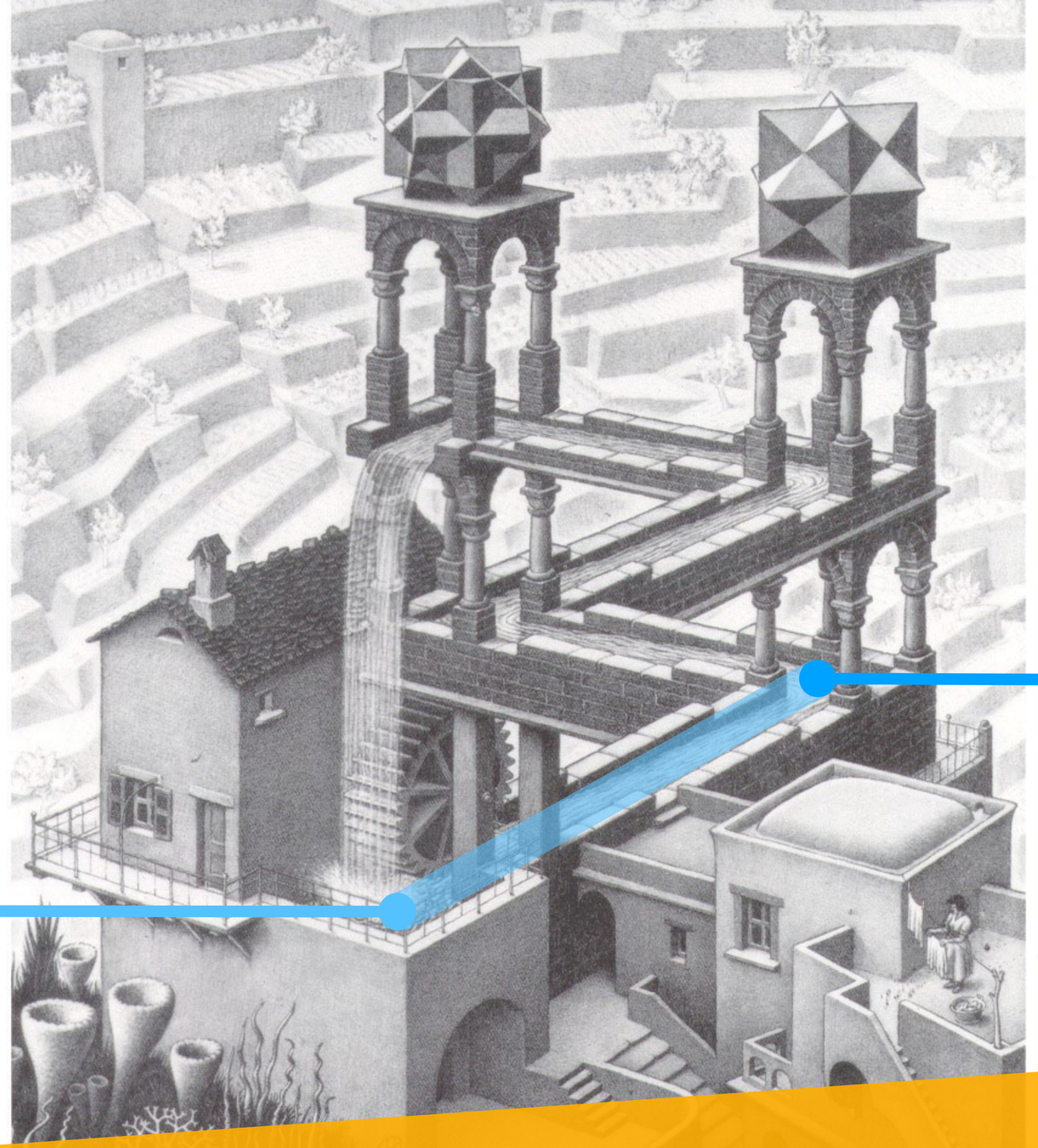


monitoring and operational support

the declarative approach

How to deal with deviations?

Outline



monitoring and operational support

the declarative approach

How to deal with multiple processes and cases?

Outline

[____,CAiSE22]

dealing with multiple processes

dealing with uncertainty

the declarative approach

dealing with multiple objects

monitoring and operational support

How to deal with multiple processes and cases?

Outline

[____,CAiSE22]

dealing with multiple proces

dealing with uncertainty

the declarative approach

Interested in uncertainty for procedural processes?

See our papers at the main track

(presentations Tue and Wed afternoon)

dealing with multiple objects

monitoring and operational support

How to deal with multiple processes and cases?

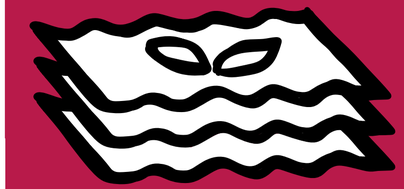
The declarative approach

Which Italian food do you like best?

Control

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

Lasagna processes



Spaghetti processes

Flexibility

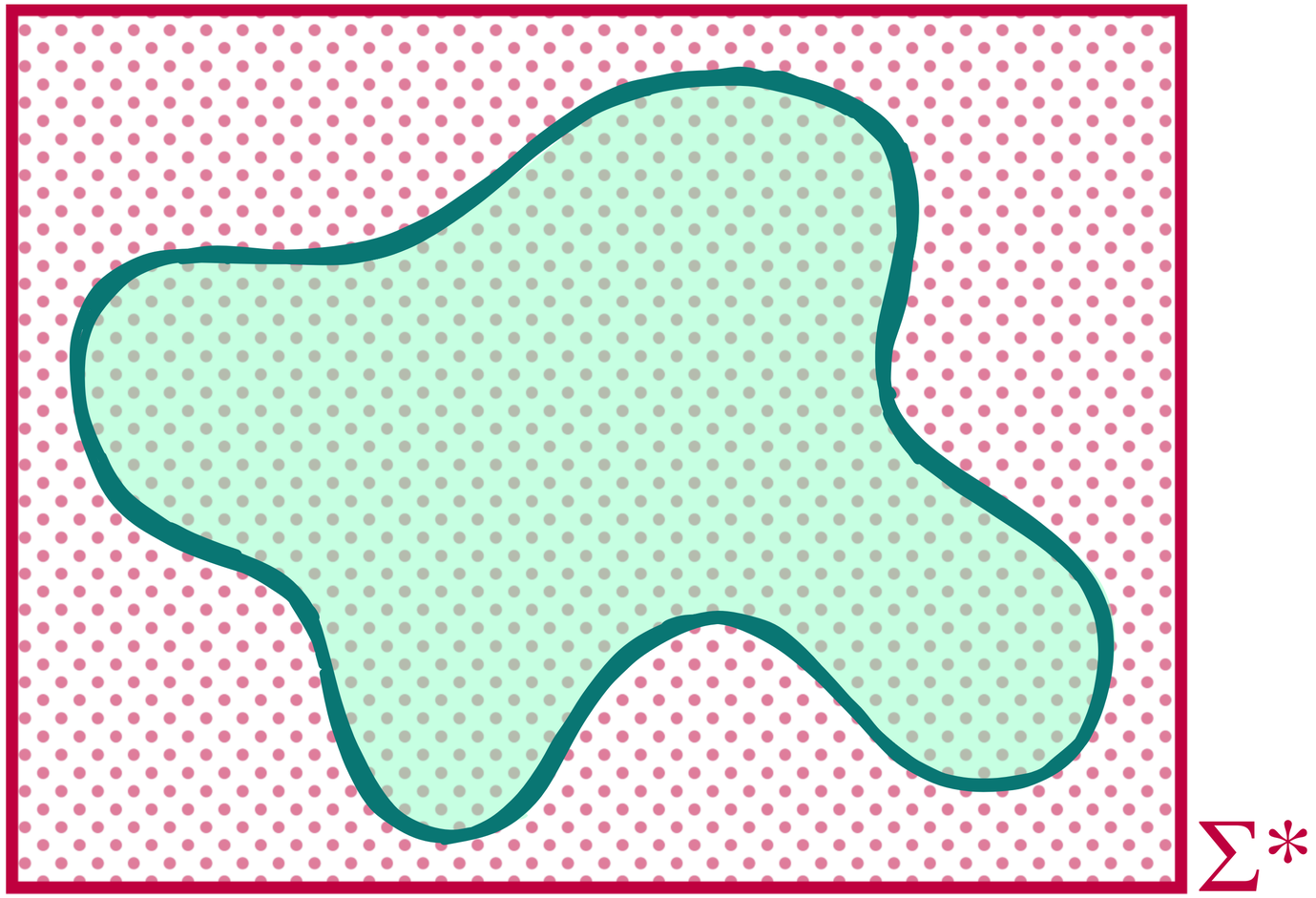
degree to which process stakeholders locally decide how to execute the process

complexity ->

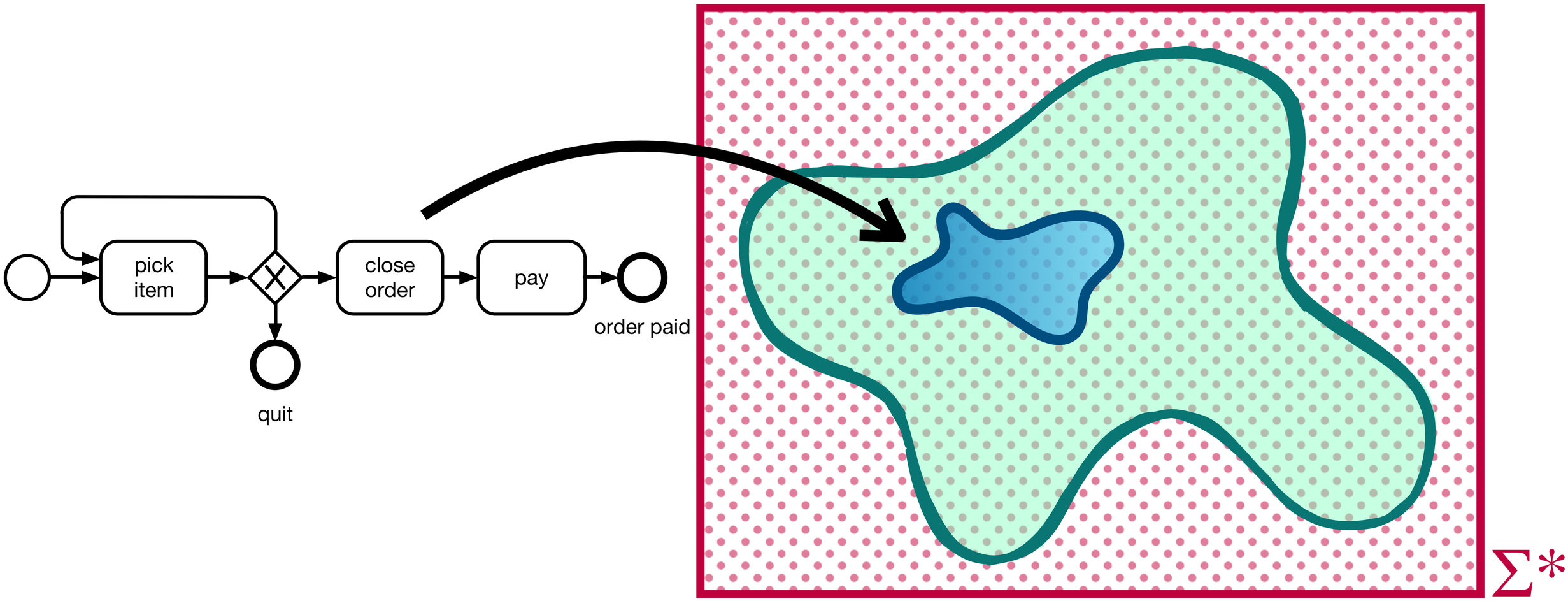
predictability <-

repetitiveness <-

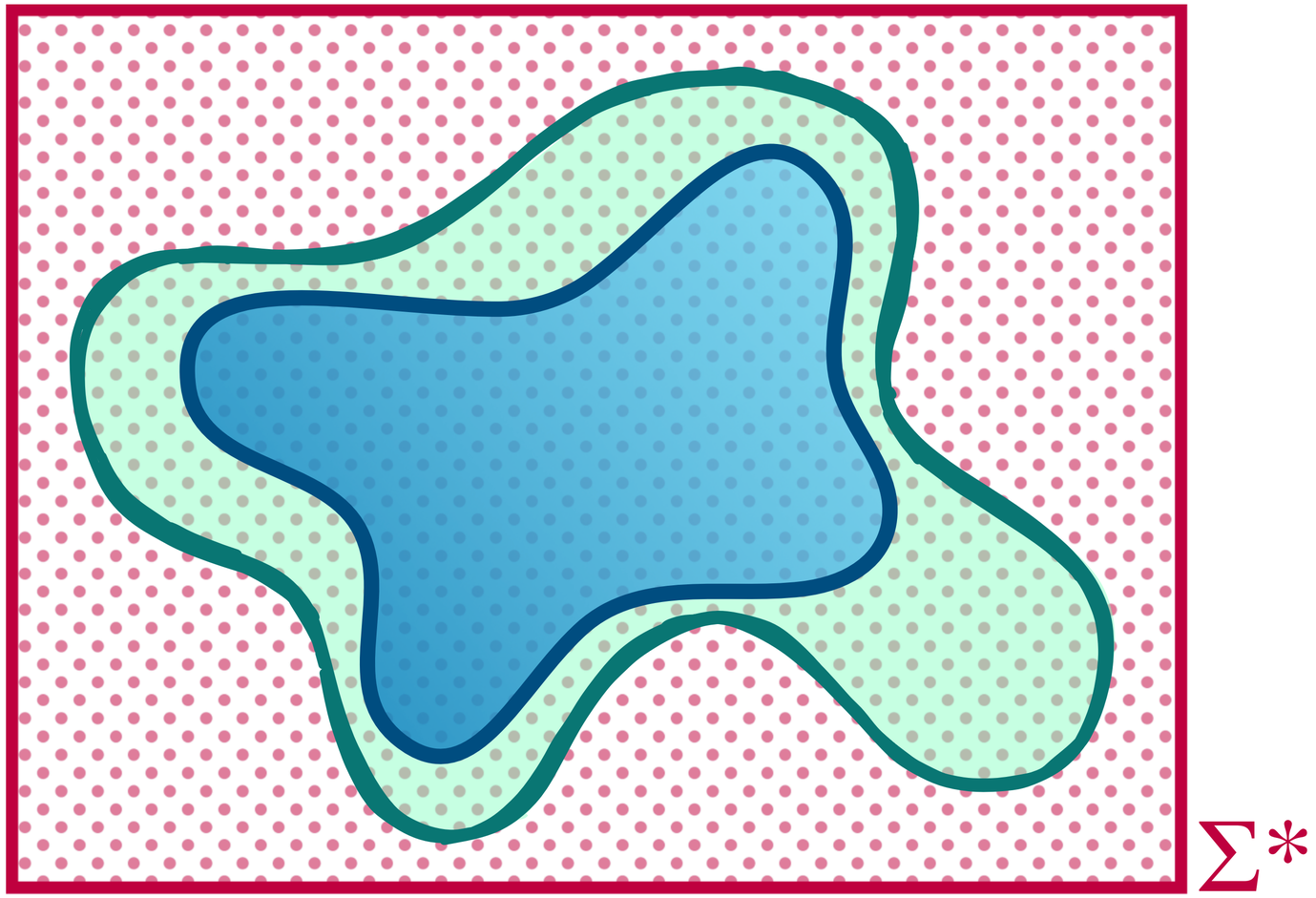
A process…



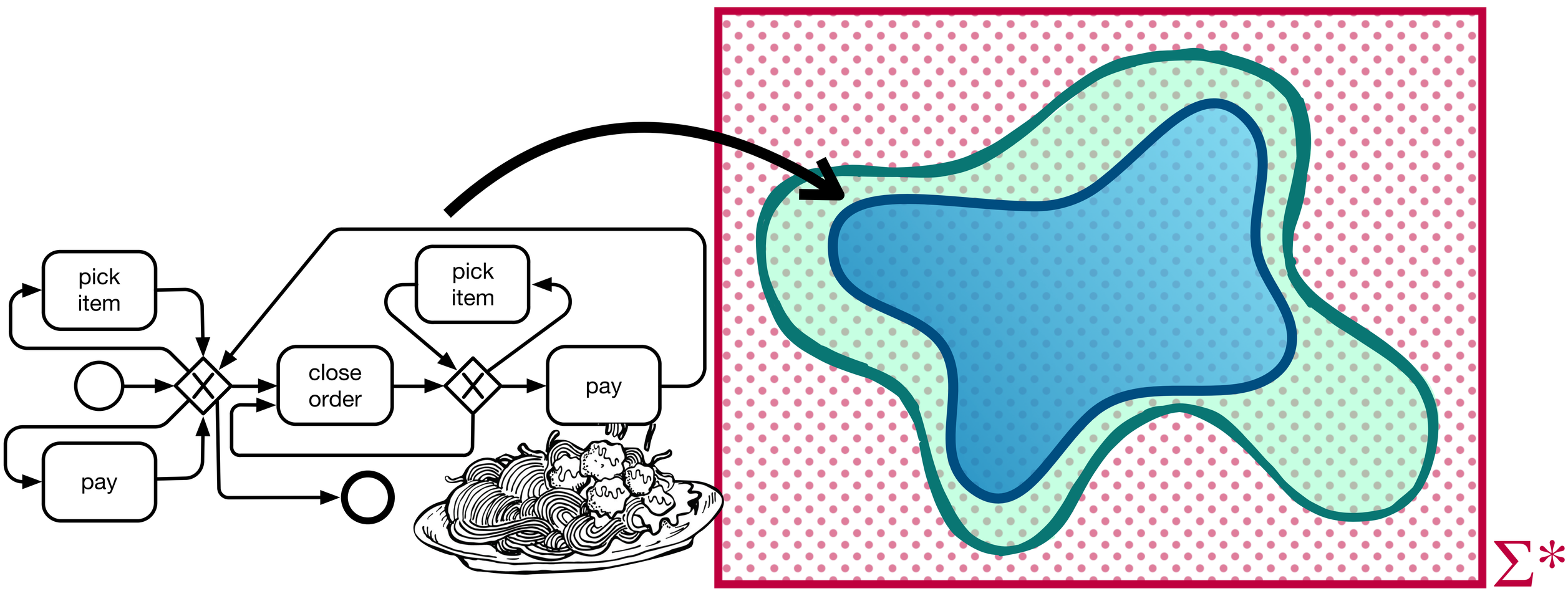
… and an imperative model of it



Generalisation



Generalisation



Simplicity cannot be obtained by sweeping complexity under the carpet

Our goal

represents

Compact specification

Reality

Our goal

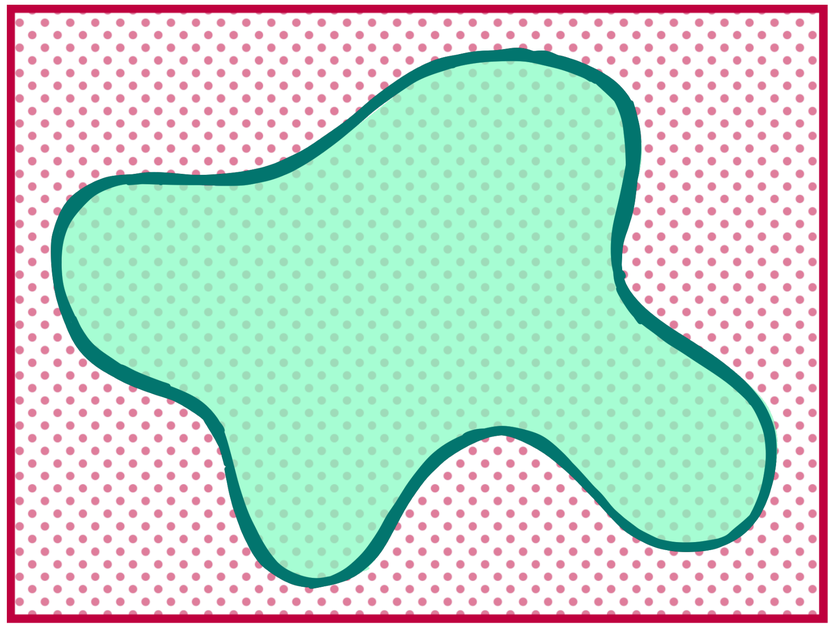
represents

The class of “regular” spaghetti processes (not all)

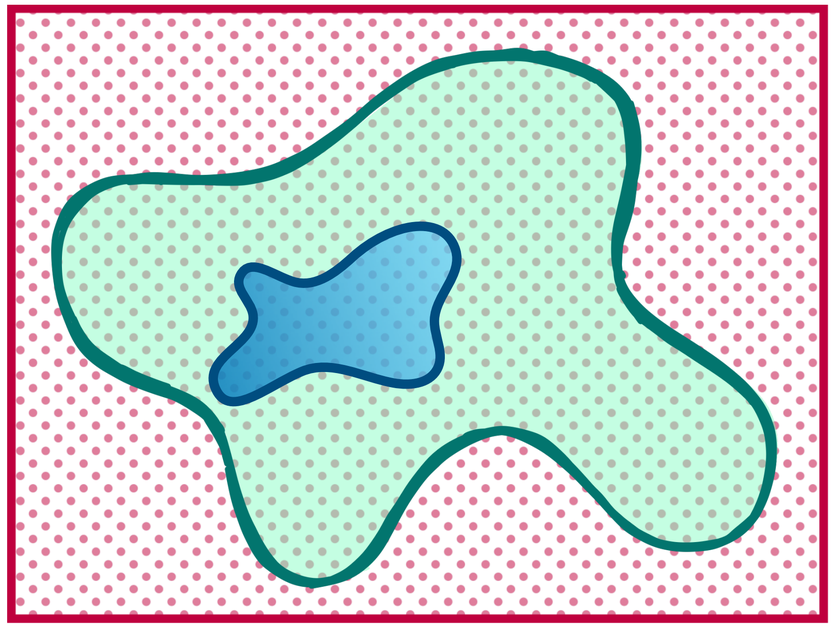
Compact specification

Reality

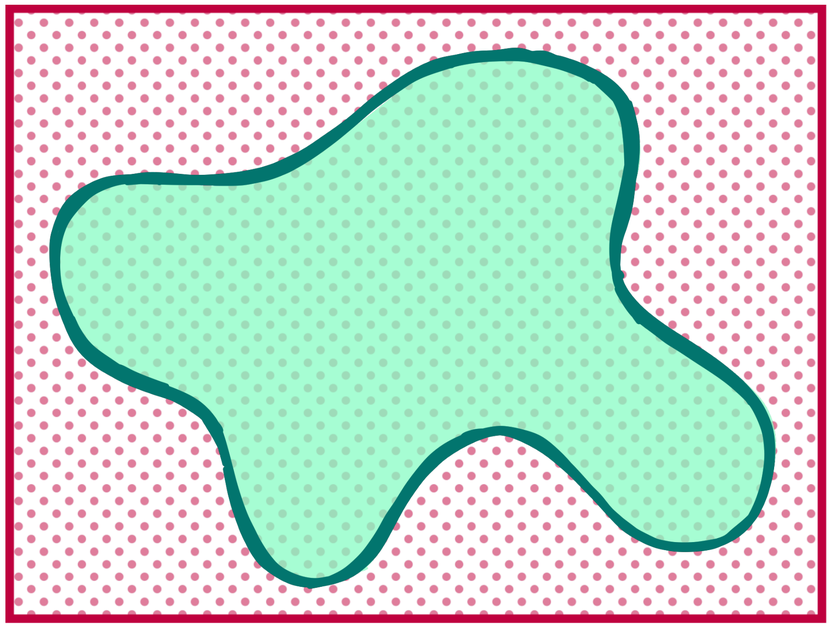
Framing via declarative specifications



Process

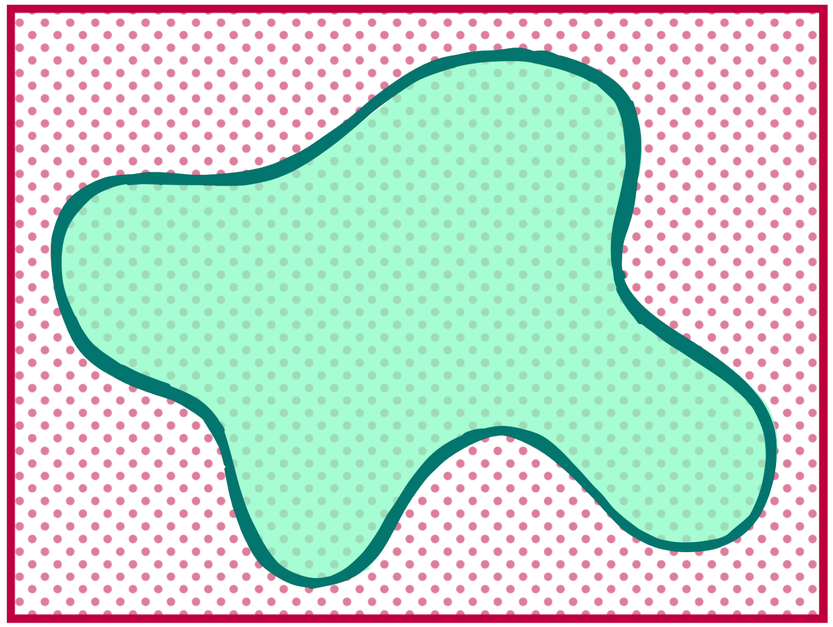


Imperative model

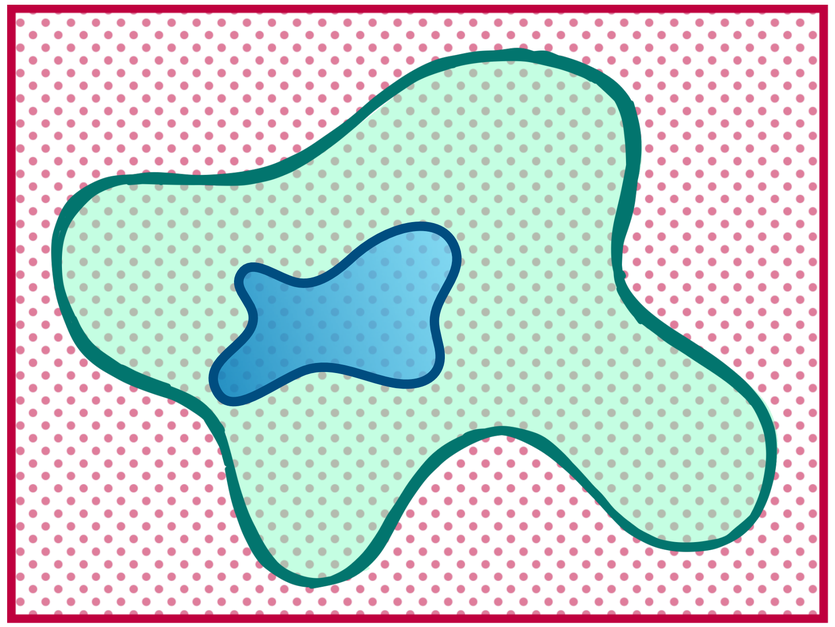


Declarative specification

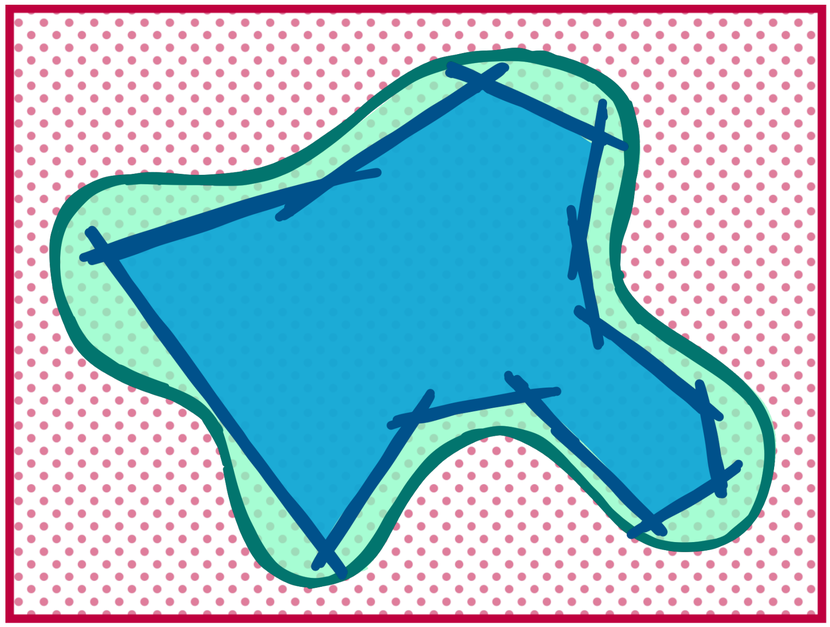
Framing via declarative specifications



Process



Imperative model



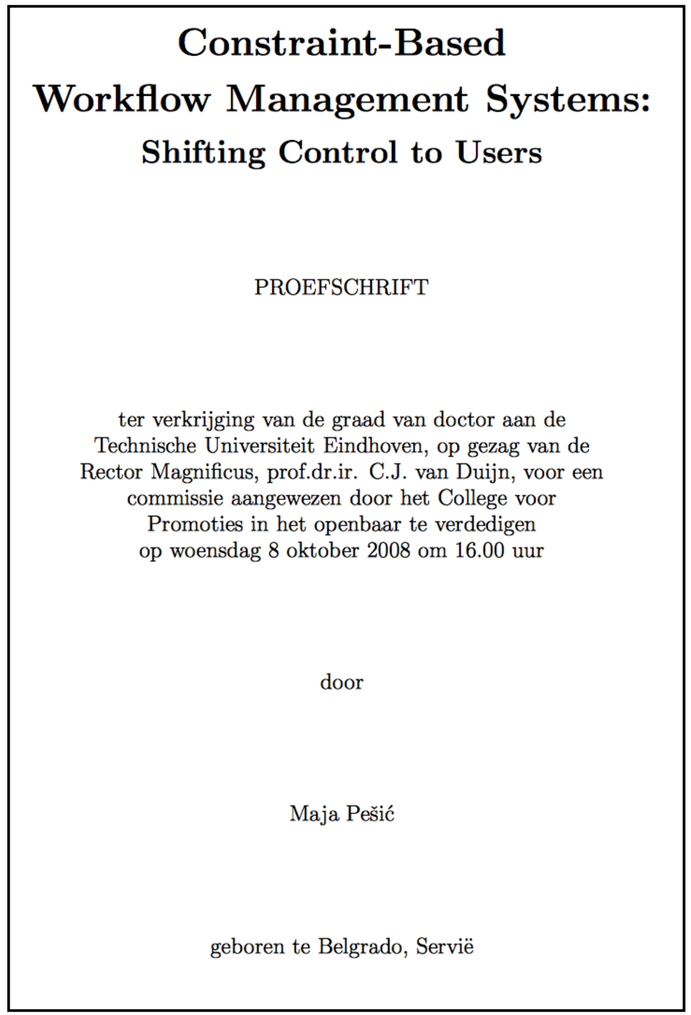
Declarative specification

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

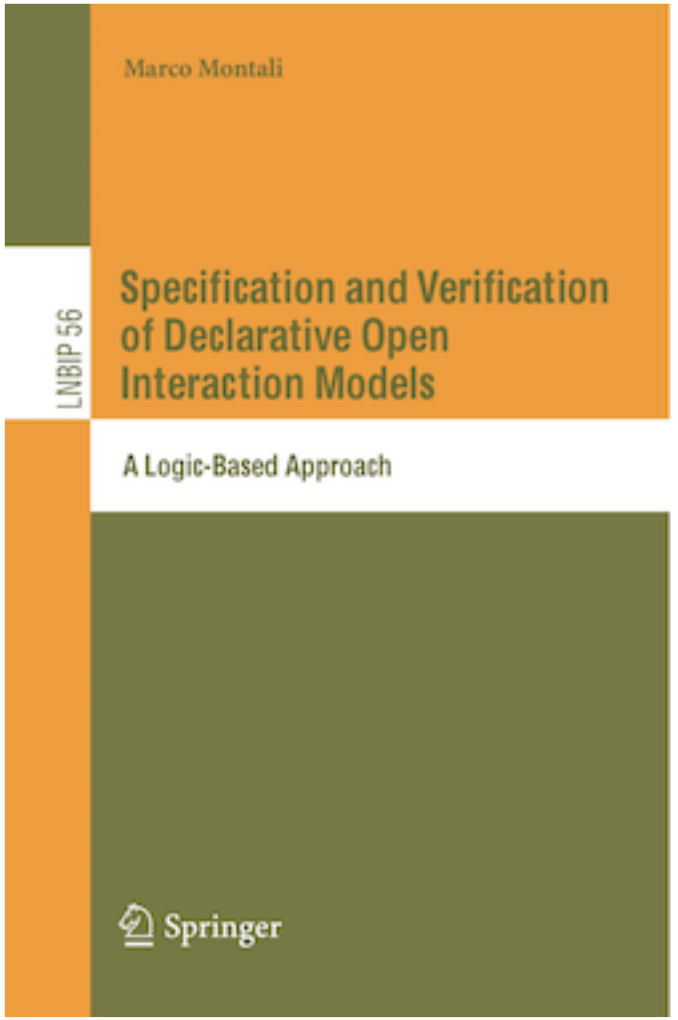
PROEFSCHRIFT

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



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

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* (when)
- **Expectation**: *negative* vs *positive*

Much richer than the precedence flow relation of imperative languages

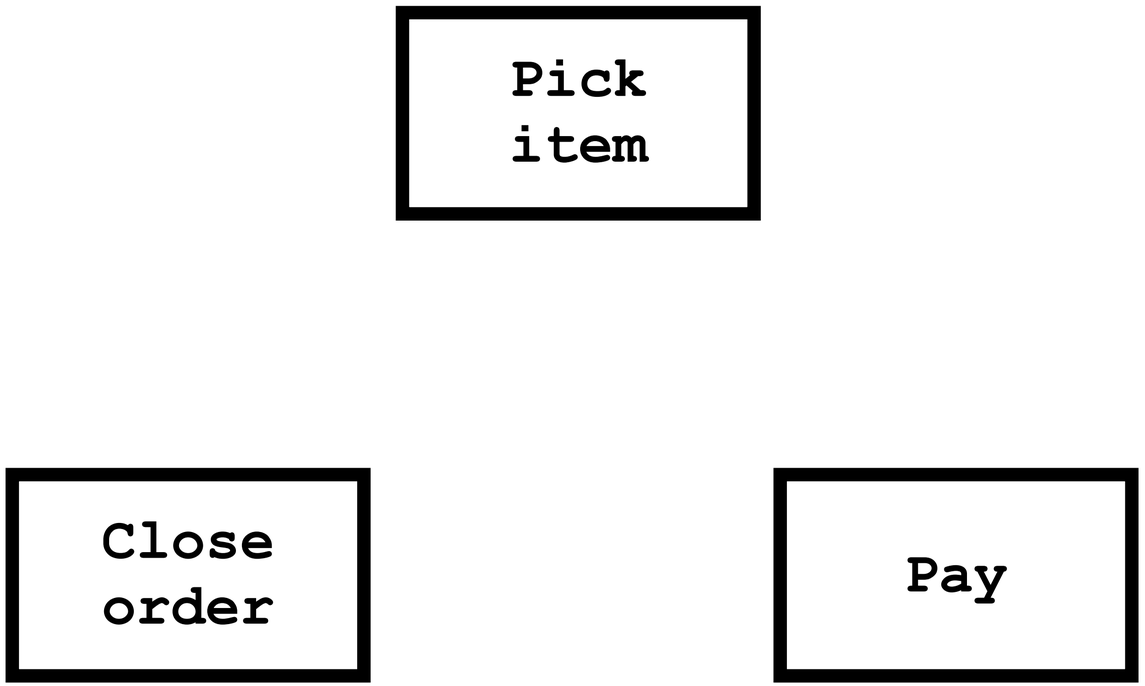
Declare specification

A set of constraints (templates grounded on the activities of interest)

- Constraints have to be **all satisfied globally** over each, **complete trace**
- Compositional approach by **conjunction**

Flexible shopper in Declare

“Whenever you close an order, you have to pay later at least once”

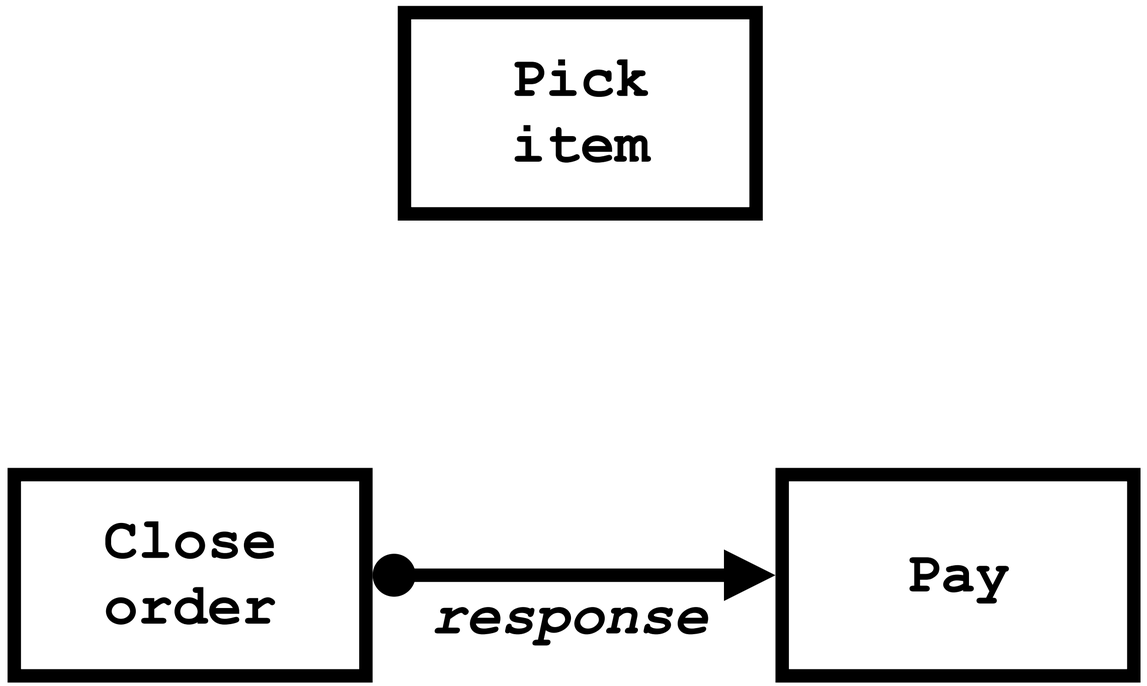


1. <> (empty trace)
2. <i,i,i>
3. <i,i,i,c,p>
4. <i,i,i,c,p,p>
5. ~~<i,i,i,p,c>~~
6. <i,c,p,i,i,c,p>

Accepts all $\{i, c, p\}^*$

Flexible shopper in Declare

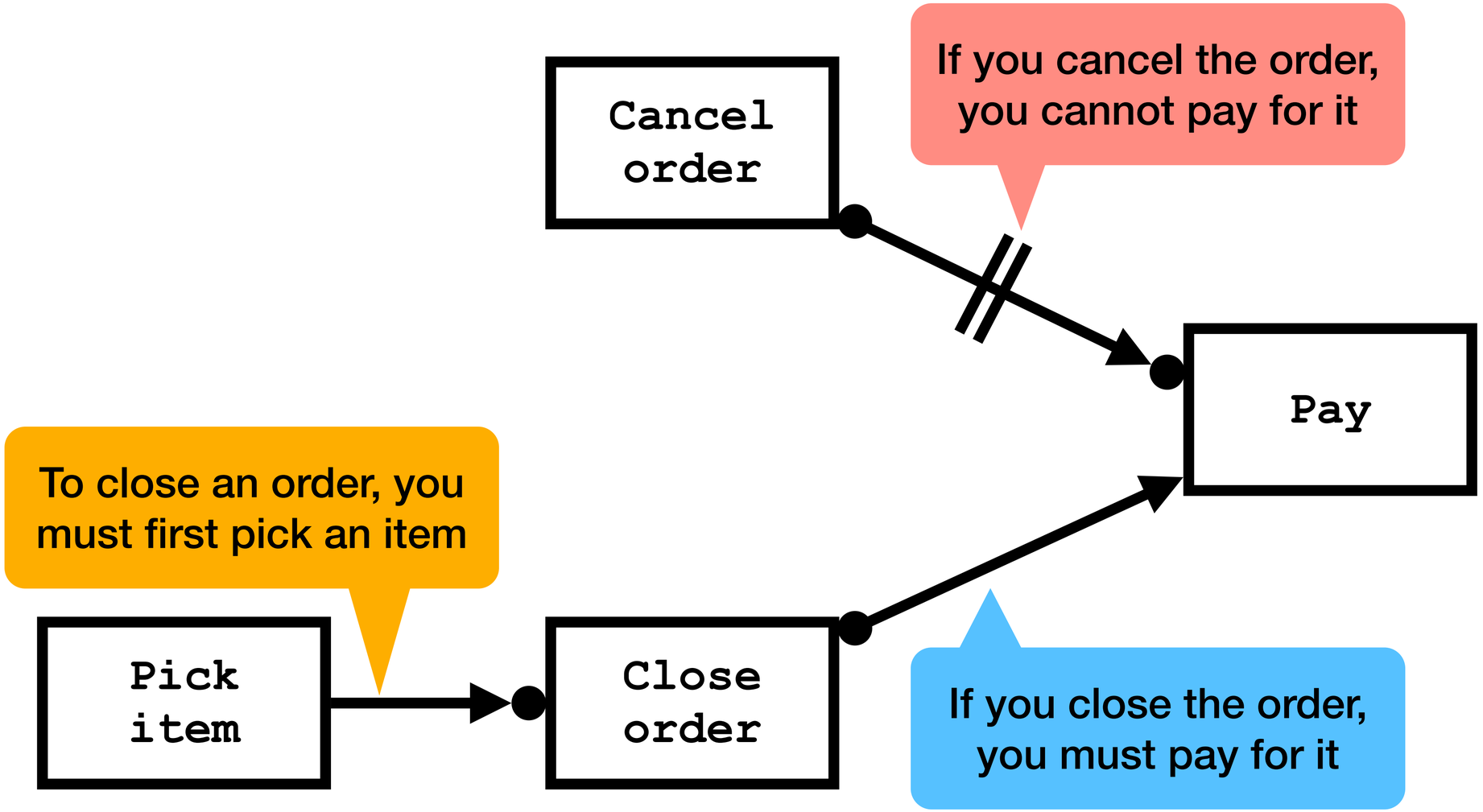
“Whenever you close an order, you have to pay later at least once”



1. <> (empty trace)
2. <i,i,i>
3. <i,i,i,c,p>
4. <i,i,i,c,p,p>
5. ~~<i,i,i,p,c>~~
6. <i,c,p,i,i,c,p>

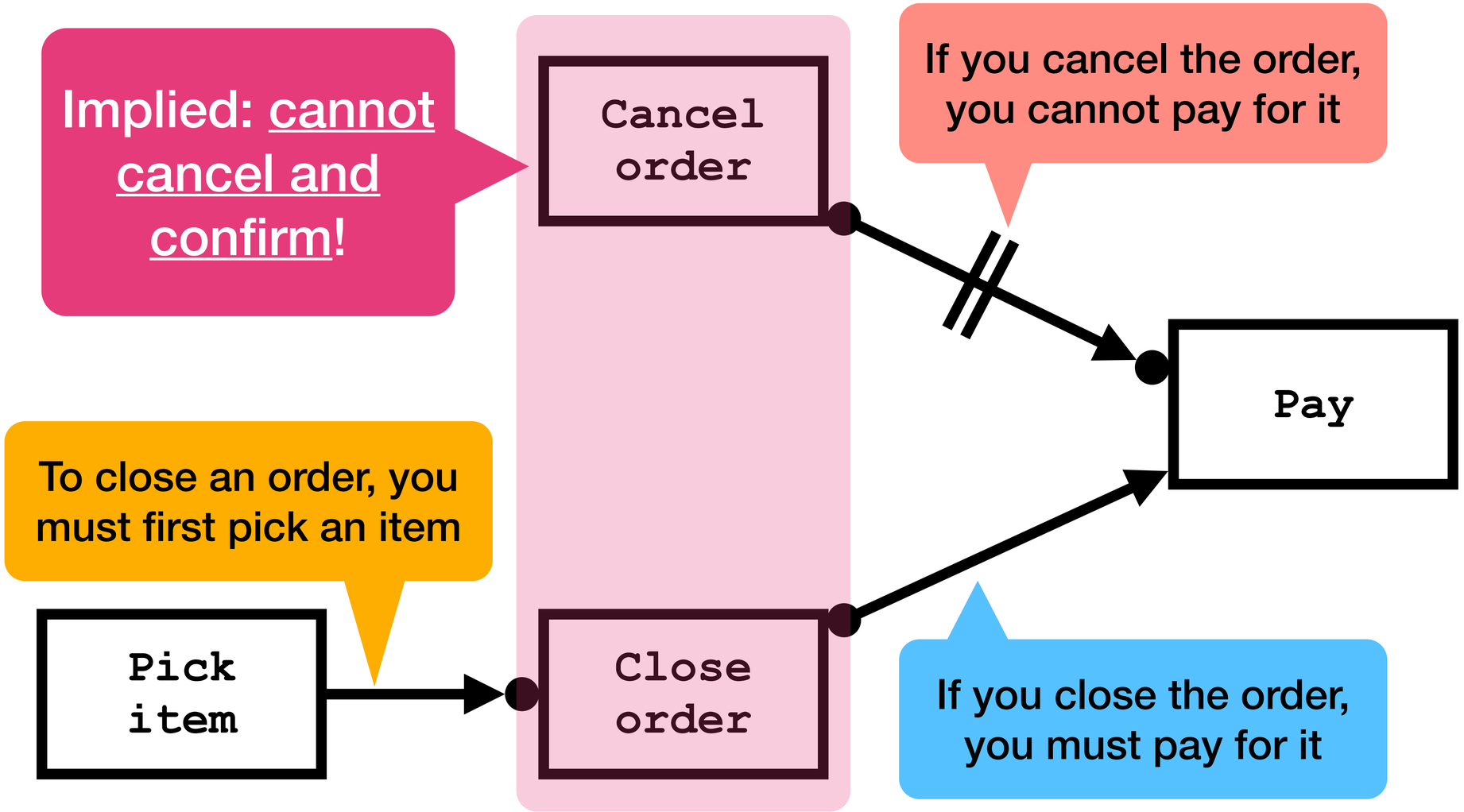
Interaction among constraints

Aka hidden dependencies [____,TWEB2010]



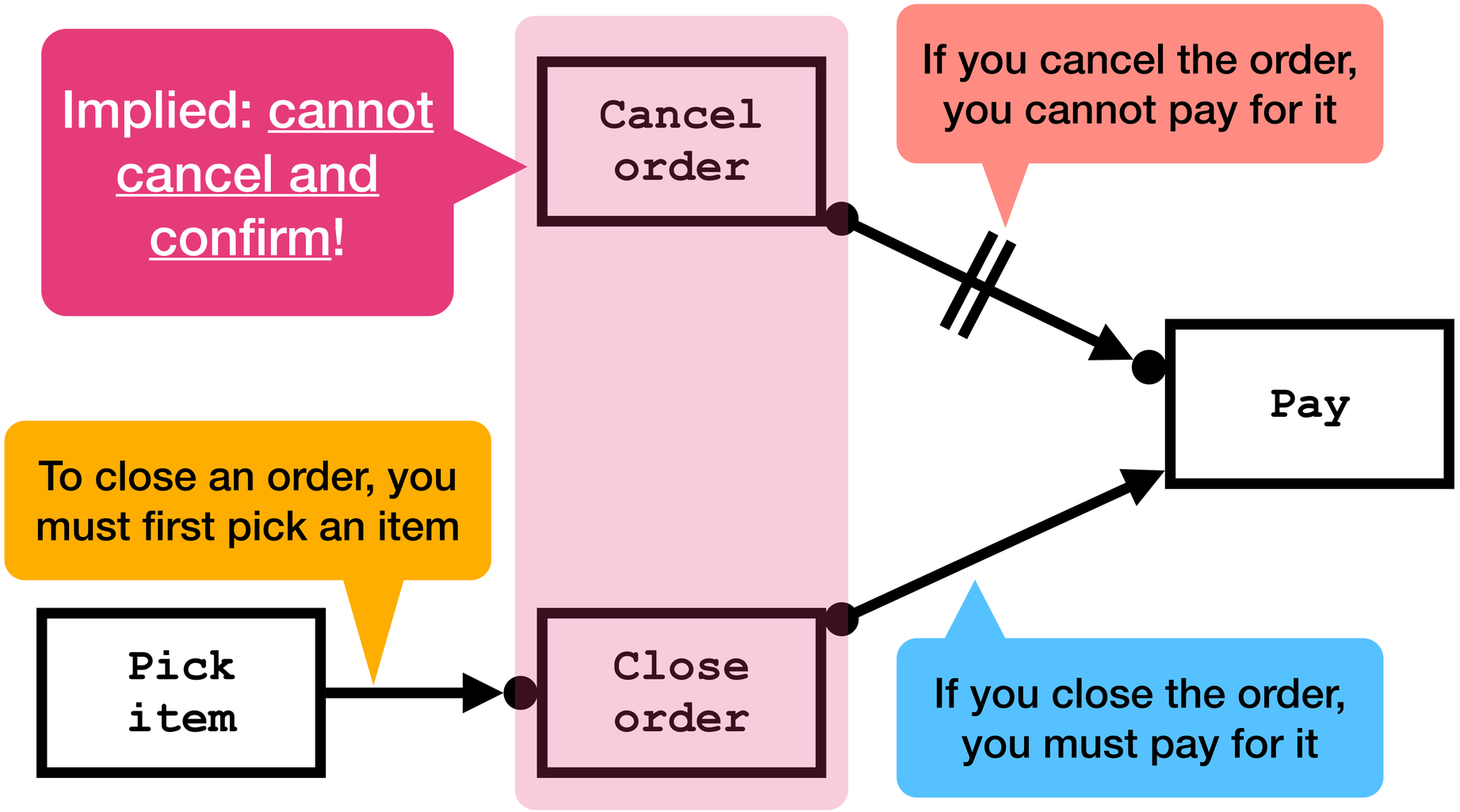
Interaction among constraints

Aka hidden dependencies [____,TWEB2010]



Interaction among constraints

Aka hidden dependencies [____,TWEB2010]



Key questions

1. How to characterise the language of a Declare specification?
2. How to understand whether a specification is correct?

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

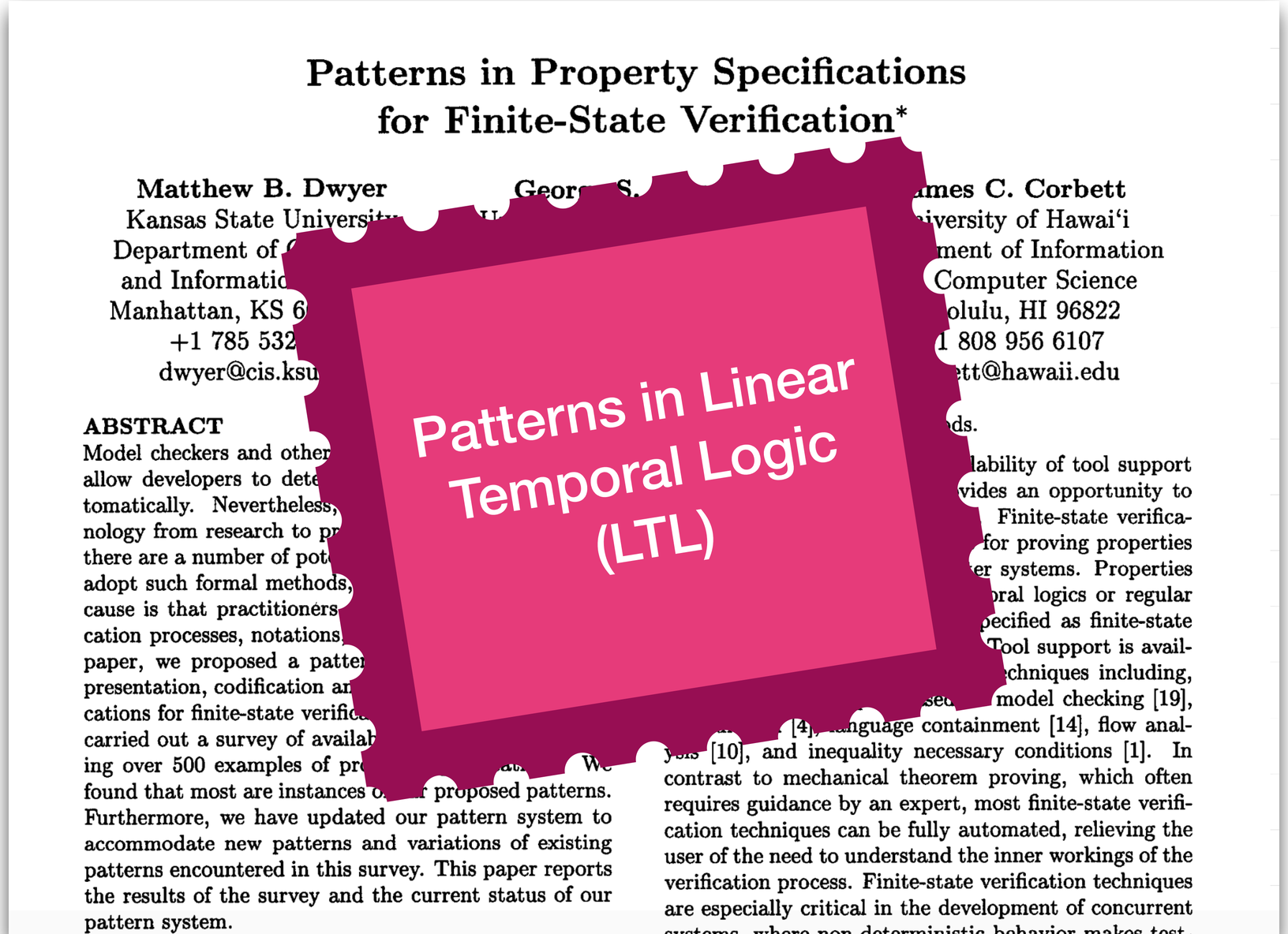
James C. Corbett
University of Hawai'i
Department of Information
and Computer Science
Honolulu, HI 96822
+1 808 956 6107
corbett@hawaii.edu

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

Back to the roots



Patterns in Property Specifications for Finite-State Verification*

Matthew B. Dwyer
Kansas State Universit
Department of
and Informatic
Manhattan, KS 6
+1 785 532
dwyer@cis.ksu

Geor S.

mes C. Corbett
iversity of Hawai'i
ment of Information
Computer Science
olulu, HI 96822
1 808 956 6107
ett@hawaii.edu

ABSTRACT
Model checkers and other
allow developers to dete
tomatically. Nevertheless,
nology from research to pr
there are a number of pot
adopt such formal methods,
cause is that practitioners
cation processes, notations,
paper, we proposed a patte
presentation, codification an
cations for finite-state verific
carried out a survey of availab
ing over 500 examples of pr
found that most are instances o 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.

ds.

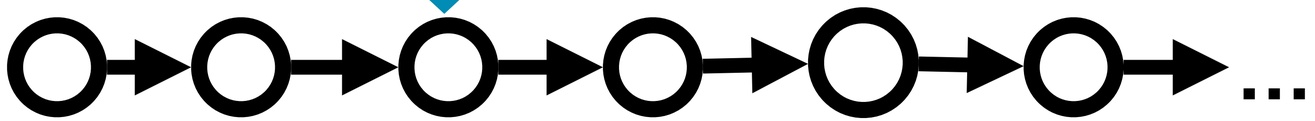
lability of tool support
vides an opportunity to
Finite-state verifica-
for proving properties
er systems. Properties
oral logics or regular
pecified as finite-state
Tool support is avail-
chniques including,
model checking [19],
[4], language containment [14], flow anal-
ysis [10], and inequality necessary conditions [1]. In
contrast to mechanical theorem proving, which often
requires guidance by an expert, most finite-state verifi-
cation 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

Patterns in Linear Temporal Logic (LTL)

LTL: a logic for infinite traces

Each state indicates which propositions hold

Logic interpreted over **infinite traces**



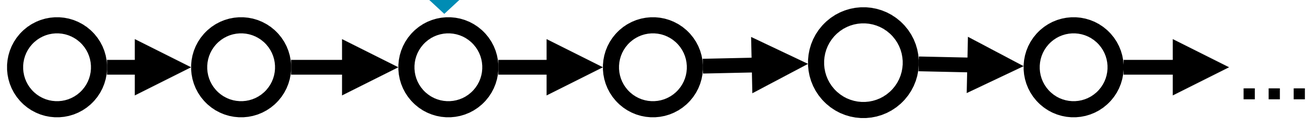
$$\varphi ::= A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

A	Atomic **propositions**
$\neg\varphi \mid \varphi_1 \wedge \varphi_2$	**Boolean connectives**
$\bigcirc\varphi$	At **next step** ϕ holds
$\varphi_1 \mathcal{U} \varphi_2$	**At some point** ϕ_2 holds, and ϕ_1 holds **until** ϕ_2 does
$\Diamond\varphi \equiv true\,\mathcal{U}\varphi$	ϕ **eventually** holds
$\Box\varphi \equiv \neg\Diamond\neg\varphi$	ϕ **always** holds
$\varphi_1 \mathcal{W} \varphi_2 \equiv \varphi_1 \mathcal{U} \varphi_2 \vee \Box\varphi_1$	ϕ_1 holds **forever or until** ϕ_2 does

LTL: a logic for infinite traces

Each state indicates which propositions hold

Logic interpreted over **infinite traces**



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

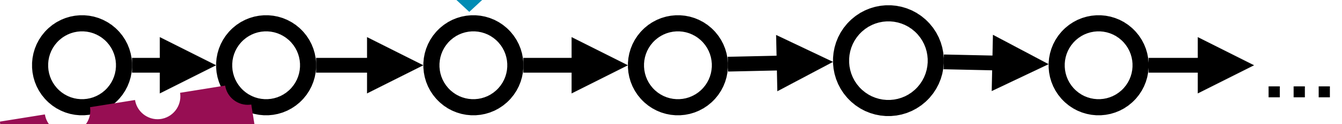
A	Atomic **propositions**
$\neg\varphi \mid \varphi_1 \wedge \varphi_2$	**Boolean connectives**
$\bigcirc\varphi$	At **next step** ϕ holds
$\varphi_1 \mathcal{U} \varphi_2$	**At some point** ϕ_2 holds, and ϕ_1 holds **until** ϕ_2 does
$\Diamond\varphi \equiv true\,\mathcal{U}\varphi$	ϕ **eventually** holds
$\Box\varphi \equiv \neg\Diamond\neg\varphi$	ϕ **always** holds
$\varphi_1 \mathcal{W} \varphi_2 \equiv \varphi_1 \mathcal{U} \varphi_2 \vee \Box\varphi_1$	ϕ_1 holds **forever or until** ϕ_2 does

Can be seamlessly extended with past-tense operators

LTL: a logic for infinite traces

Each state indicates which propositions hold

Logic interpreted over **infinite traces**



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

A	
$\neg\varphi \mid \varphi_1 \wedge \varphi_2$	
$\bigcirc\varphi$	
$\varphi_1 \mathcal{U} \varphi_2$	ϕ_1 holds **until** ϕ_2 does
$\Diamond\varphi \equiv true\,\mathcal{U}\varphi$	
$\Box\varphi \equiv \neg\Diamond\neg\varphi$	
$\varphi_1 \mathcal{W} \varphi_2 \equiv \varphi_1 \mathcal{U} \varphi_2 \vee \Box\varphi_1$	ϕ_1 holds **forever or until** ϕ_2 does

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

$$t \vDash \varphi$$

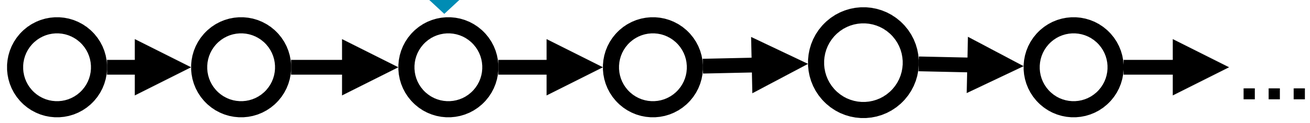
Can be seamlessly extended with past-tense operators

Template formulae

Template	**LTL$_f$ expression**
Existence constraints	
$\textsc{AtLeastOne}(x)$	$\Box\,(\mathbf{start} \rightarrow \Diamond\, x)$
$\textsc{AtMostOne}(x)$	$\Box(x \rightarrow \neg \bigcirc \Diamond\, x)$
$\textsc{Init}(x)$	$\Box\,(\mathbf{start} \rightarrow x)$
$\textsc{End}(x)$	$\Box\,(\mathbf{end} \rightarrow x)$
Relation constraints	
$\textsc{RespondedExistence}(x, y)$	$\Box\,(x \rightarrow \Diamond\, y \vee \Diamond\!\!-\, y)$
$\textsc{Response}(x, y)$	$\Box\,(x \rightarrow \Diamond\, y)$
$\textsc{AlternateResponse}(x, y)$	$\Box\,(x \rightarrow \bigcirc(\neg x\ \mathbf{U}\ y))$
$\textsc{ChainResponse}(x, y)$	$\Box\,(x \rightarrow \bigcirc\, y)$
$\textsc{Precedence}(x, y)$	$\Box\,(y \rightarrow \Diamond\!\!-\, x)$
$\textsc{AlternatePrecedence}(x, y)$	$\Box\,(y \rightarrow \ominus(\neg y\ \mathbf{S}\ x))$
$\textsc{ChainPrecedence}(x, y)$	$\Box\,(y \rightarrow \ominus\, x)$
Negative relation constraints	
$\textsc{NotRespondedExistence}(x, y)$	$\Box(x \rightarrow (\Box\, \neg y \wedge \boxminus\, \neg y))$
$\textsc{NotResponse}(x, y)$	$\Box(x \rightarrow \Box\, \neg y)$
$\textsc{NotChainResponse}(x, y)$	$\Box(x \rightarrow \neg \bigcirc\, y)$
$\textsc{NotPrecedence}(x, y)$	$\Box(y \rightarrow \boxminus\, \neg x)$
$\textsc{NotChainPrecedence}(y, x)$	$\Box(y \rightarrow \neg \ominus\, x)$

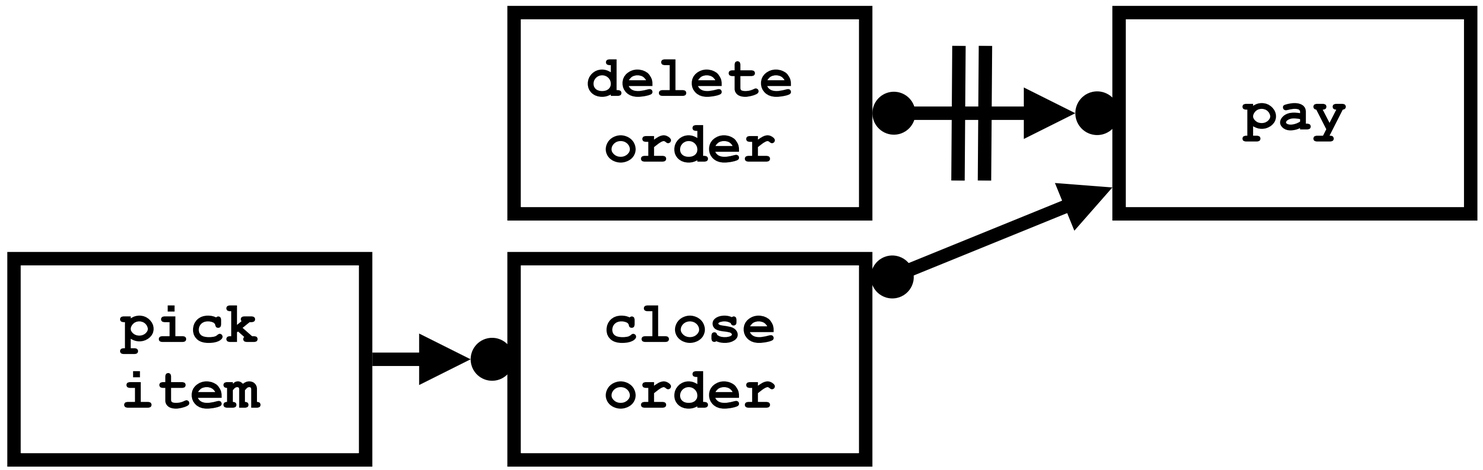
Semantics of Declare via LTL

Each state contains a single activity



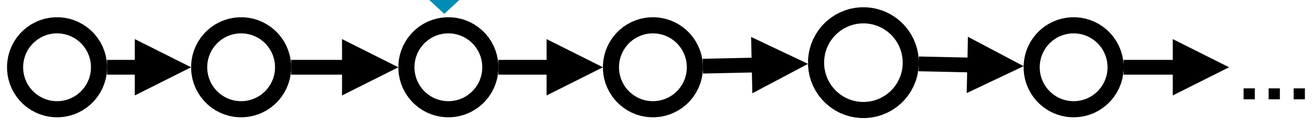
Atomic **propositions** are **activities**

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



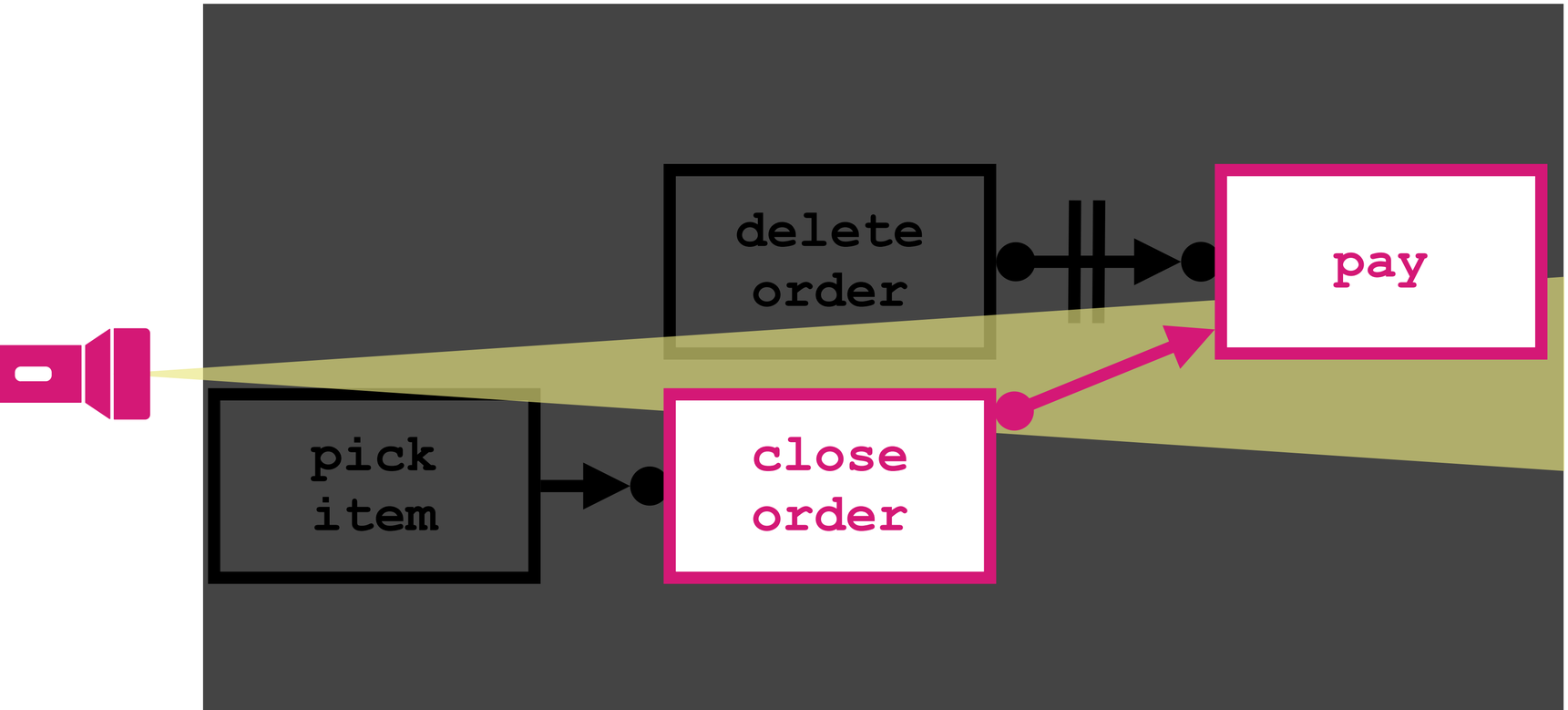
Semantics of Declare via LTL

Each state contains a single activity



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

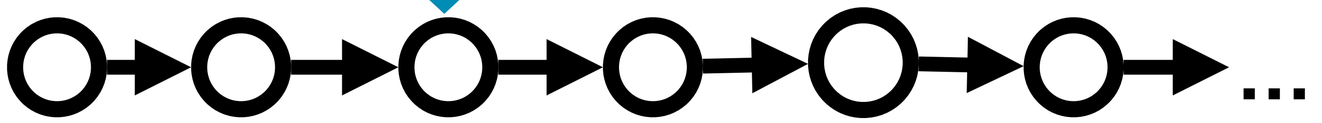
Each **constraint** is an **LTL formula** (built from template formulae)



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

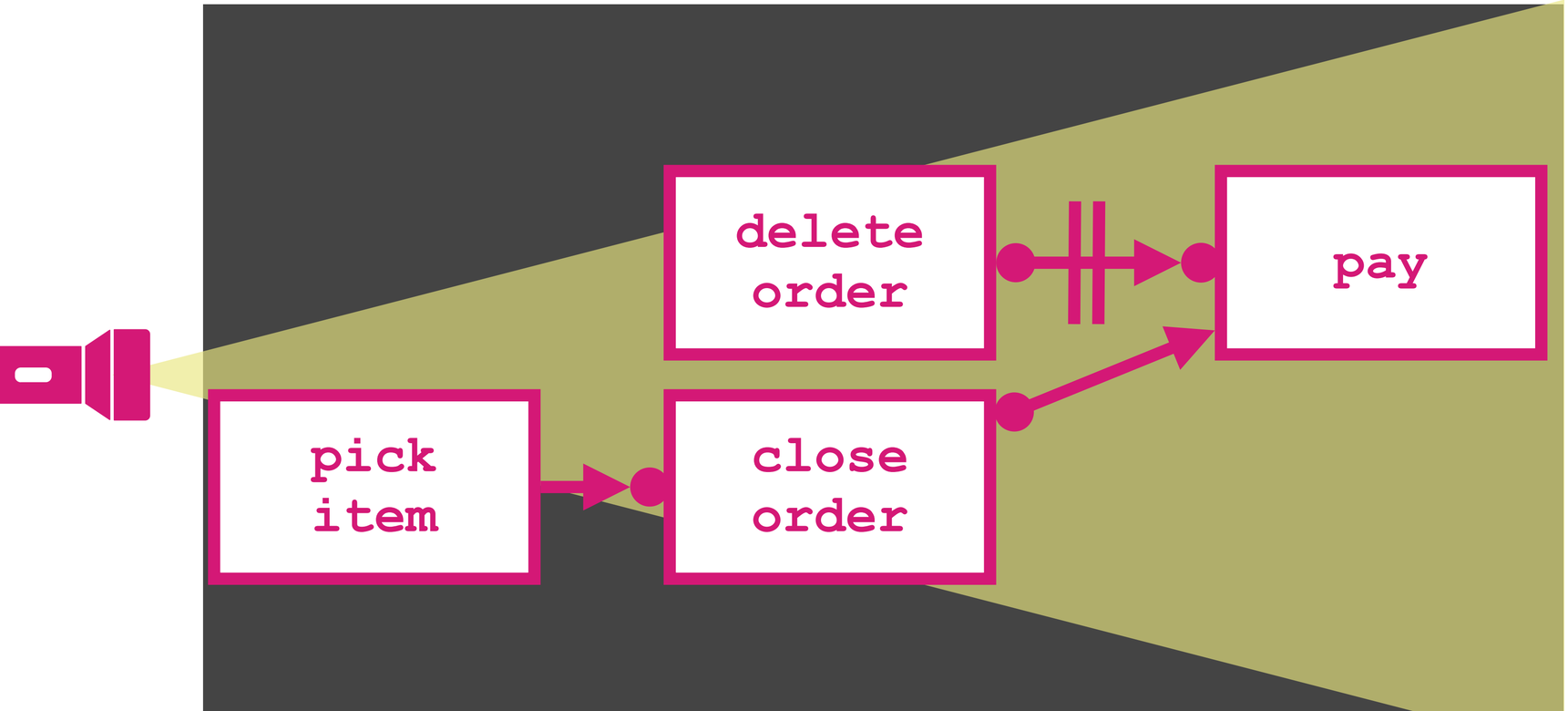
Semantics of Declare via LTL

Each state contains a single activity



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

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



$$\Box(close \rightarrow \Diamond pay)$$
$$\wedge \Box(close \rightarrow \Diamond item)$$
$$\wedge \Box(cancel \rightarrow \neg \Box pay)$$

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**

Wait a moment…


□(close → ◇pay)
Close order
Just what we needed!
But recall: each process instance should complete in a finite number of steps!
Wait a moment…

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

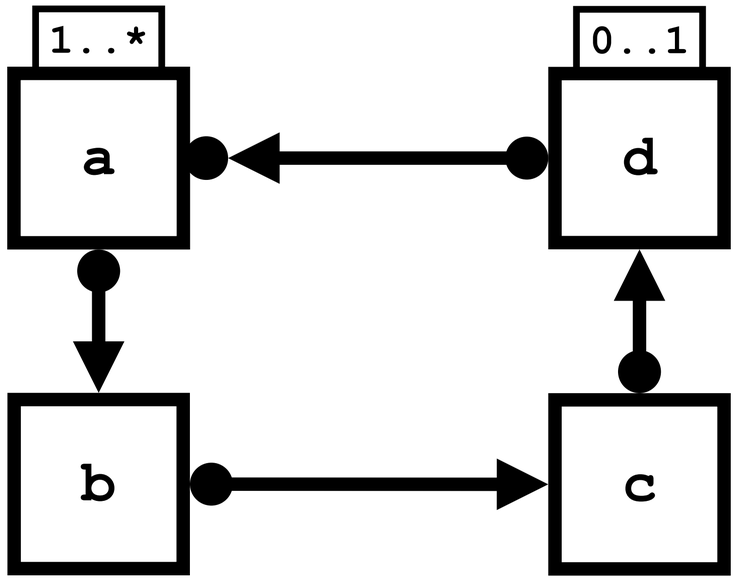
Look the same, but they are not the same

Many researchers: **misled by moving from infinite to finite traces**

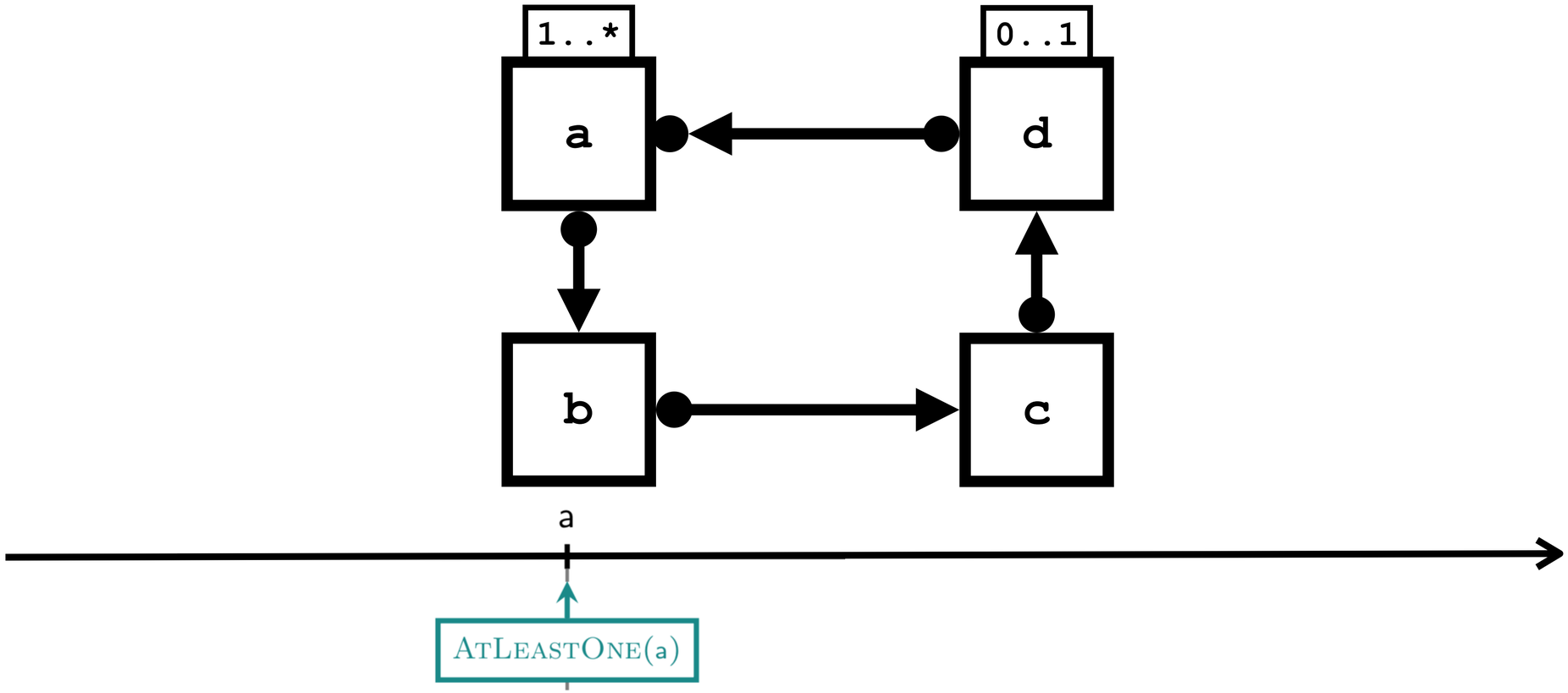
In **[____,AAAI14]**, we studied why!

- People typically focus on “patterns”, not on the entire logic
- Many of such patterns in BPM, reasoning about actions, planning, etc. are “insensitive to infinity”

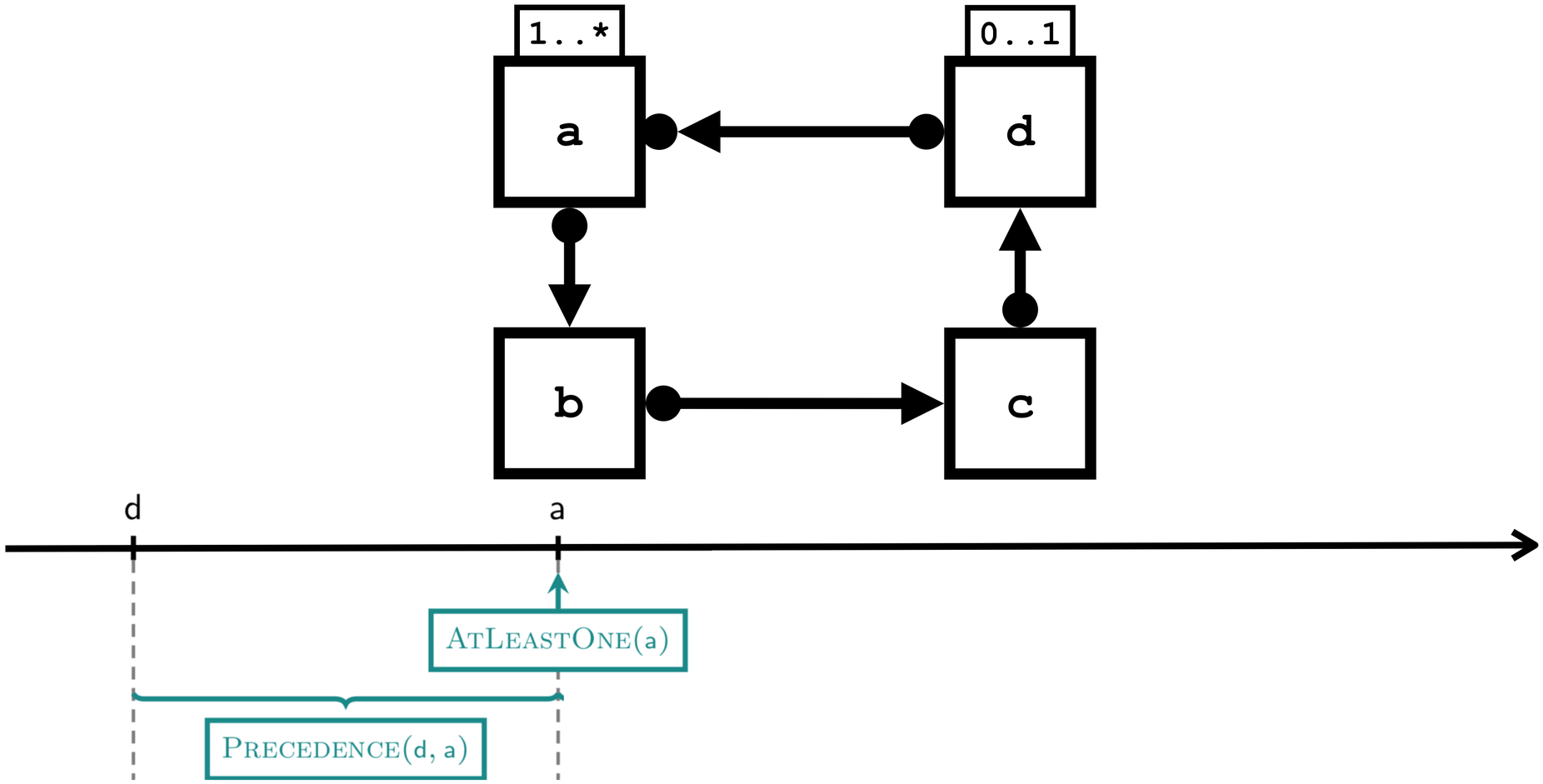
Quiz: does this specification accept traces?



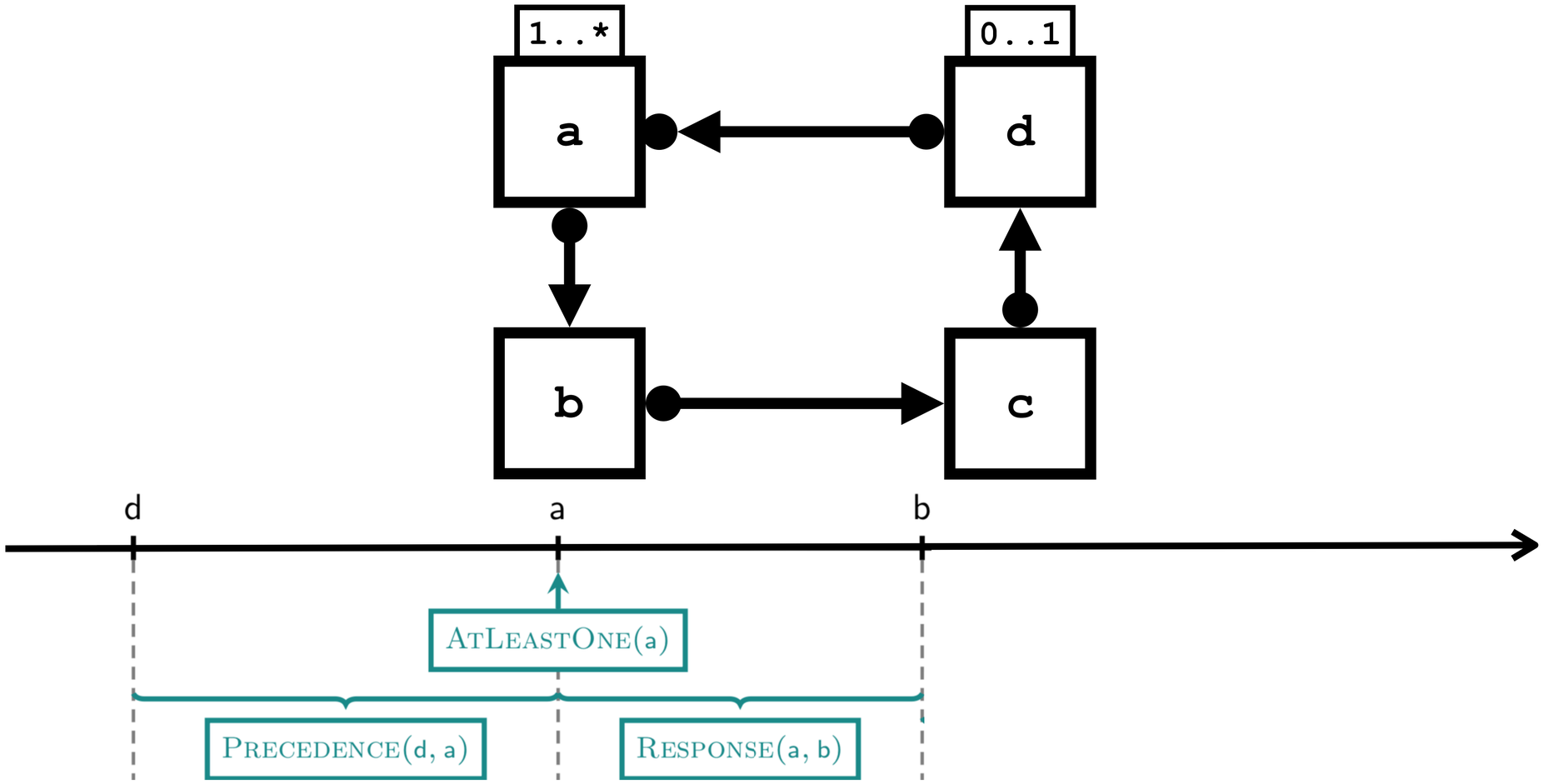
Quiz: does this specification accept traces?



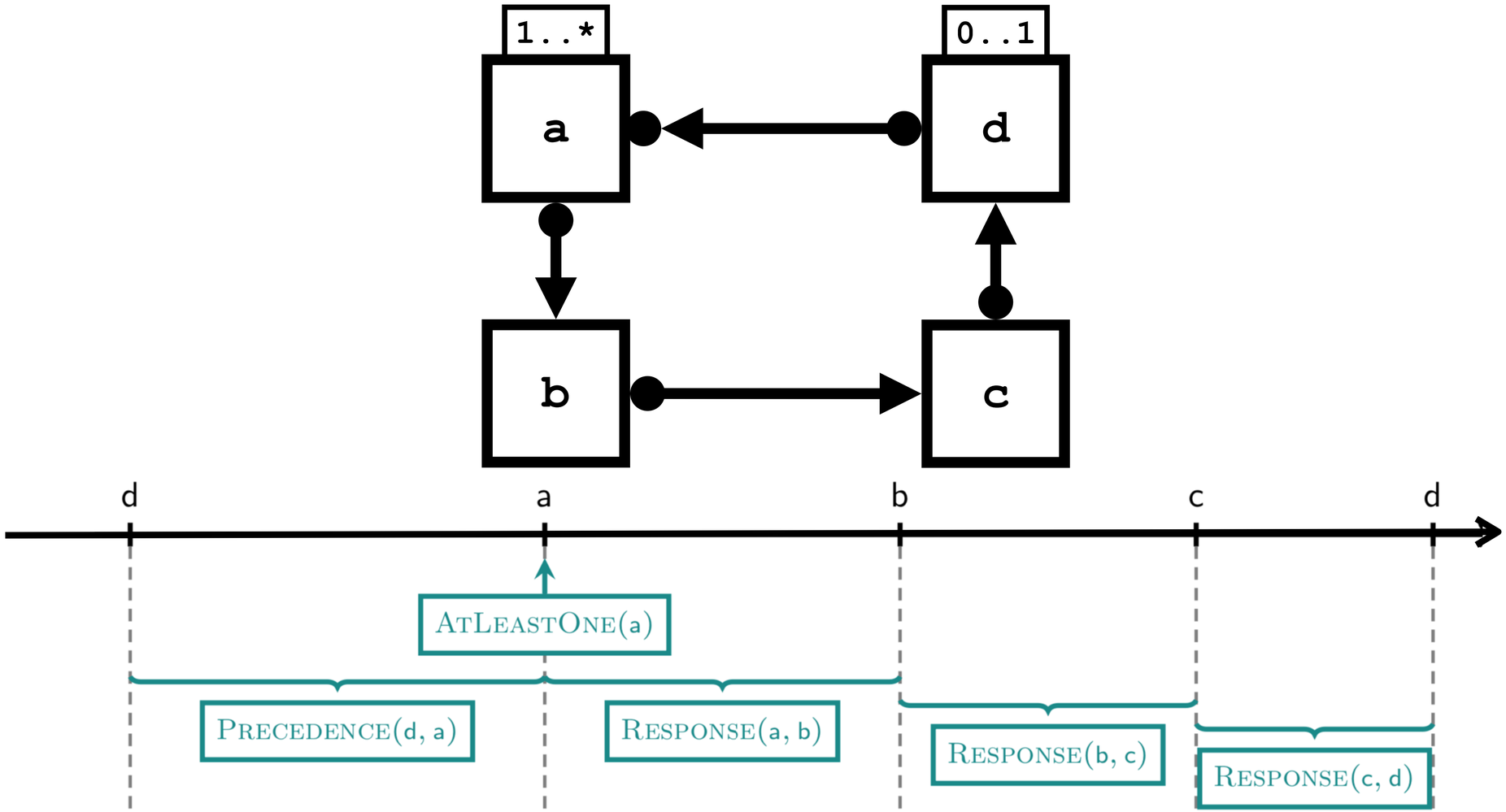
Quiz: does this specification accept traces?



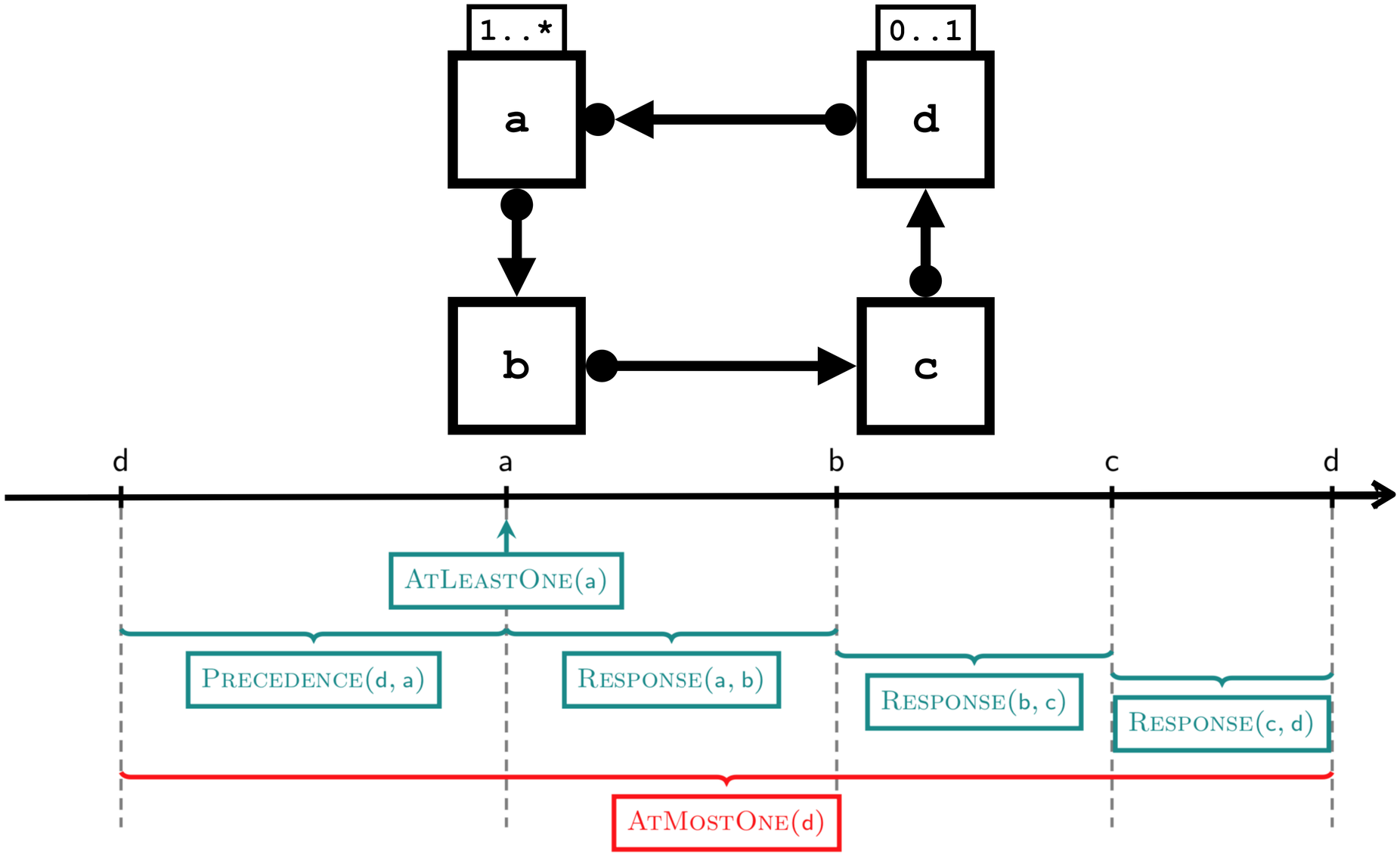
Quiz: does this specification accept traces?



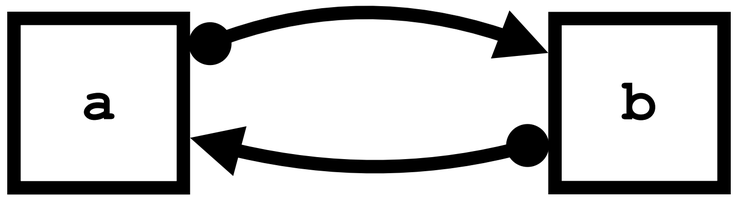
Quiz: does this specification accept traces?



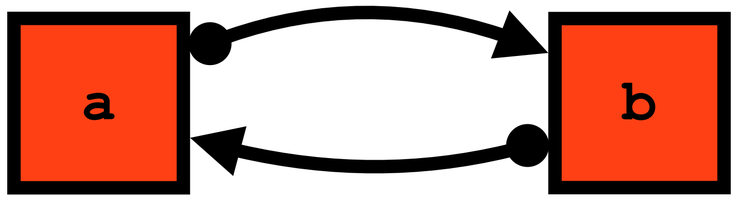
Quiz: does this specification accept traces?



Quiz: does this specification accept traces?



Quiz: does this specification accept traces?



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

How to do this, systematically?

Declare specification: encoded in **LTLf**

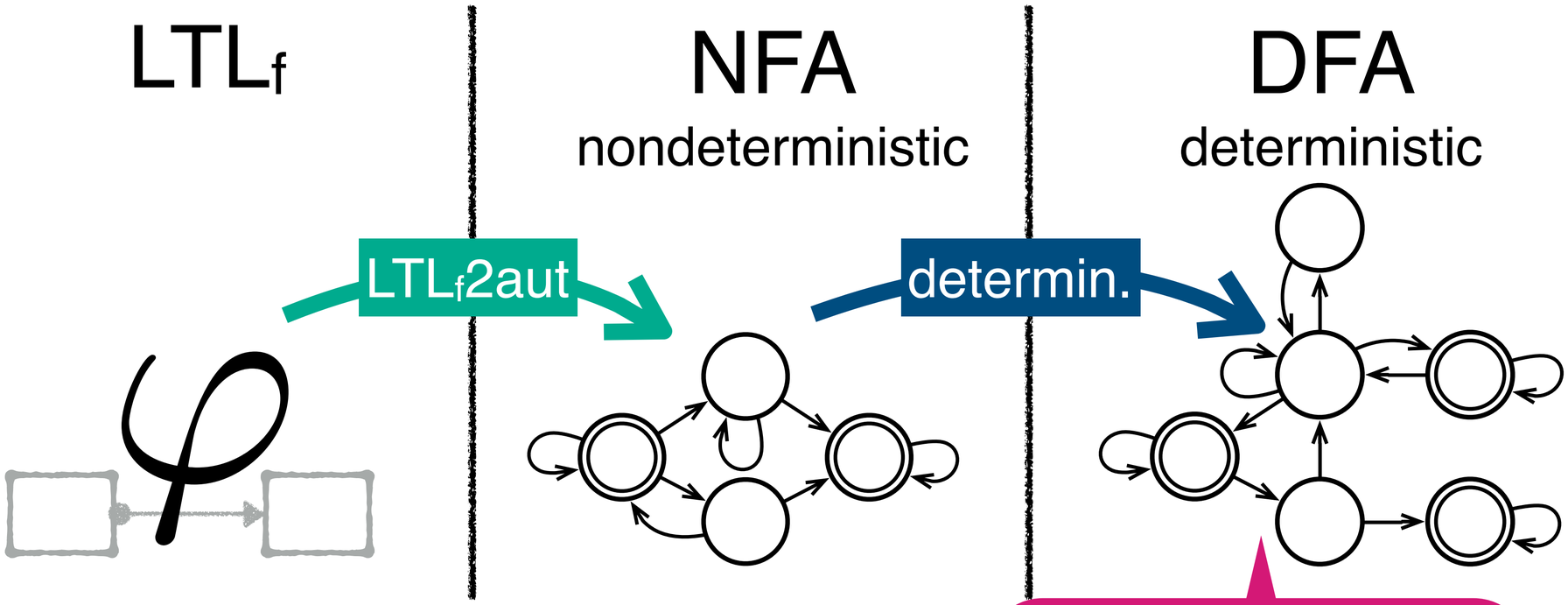
LTLf: the star-free **fragment of regular expressions**

Regular expressions: intimately connected to **finite-state automata**

- **No exotic automata models over infinite structures!**
- Just the **good-old deterministic finite-state automata (DFAs)** you know from a typical course in *programming languages and compilers*

From Declare to automata

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



[DeGiacomoVardi,IJCAI2013]
[____,TOSEM2022]

Process engine!

Vision realised!

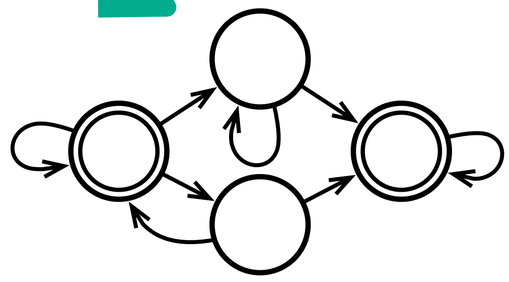
$$t \models \varphi \text{ iff } t \in \mathcal{L}(\mathcal{A}_\varphi)$$

NFA
nondeterministic

DFA
deterministic

LTL$_f$2aut

determin.

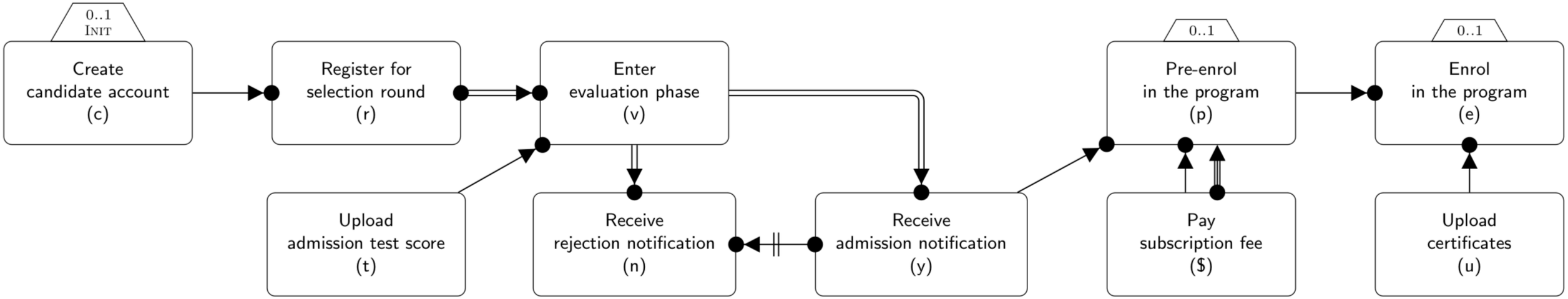


[DeGiacomoVardi,IJCAI2013]
[____,TOSEM2022]

Process engine!

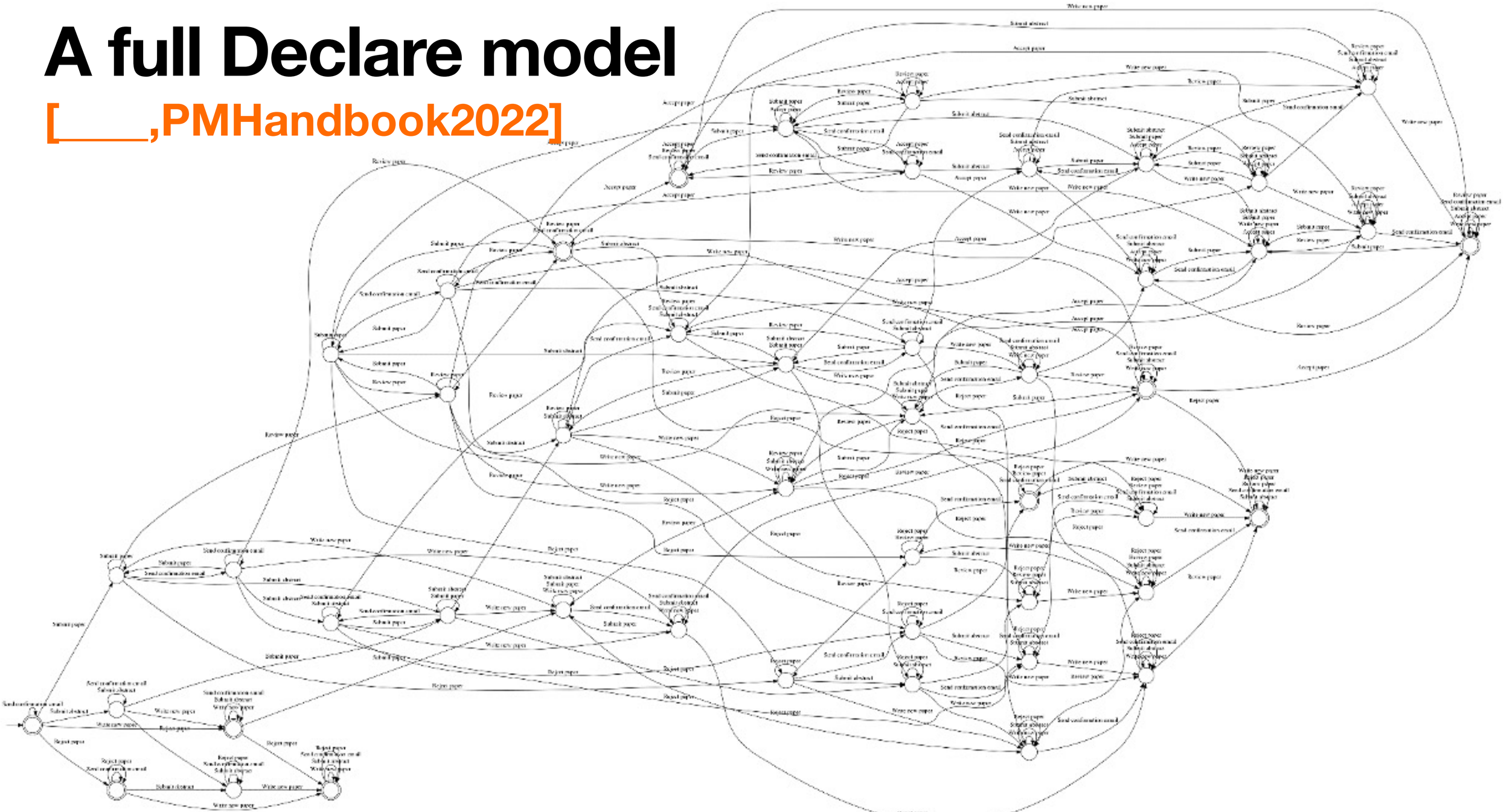
A full Declare model

[____,PMHandbook2022]



A full Declare model

[____,PMHandbook2022]



Few lines of code

[____,TOSEM2022]

$$\varphi \rightarrow$$

$$\begin{aligned}
\delta(tt,\Pi) &= true\\
\delta(ff,\Pi) &= false\\
\delta(\phi,\Pi) &= \delta(\langle\phi\rangle tt,\Pi) \quad (\phi \text{ prop.})\\
\delta(\varphi_1 \wedge \varphi_2,\Pi) &= \delta(\varphi_1,\Pi) \wedge \delta(\varphi_2,\Pi)\\
\delta(\varphi_1 \vee \varphi_2,\Pi) &= \delta(\varphi_1,\Pi) \vee \delta(\varphi_2,\Pi)\\
\delta(\langle\phi\rangle\varphi,\Pi) &= \begin{cases} E(\varphi) \text{ if } \Pi \models \phi \quad (\phi \text{ prop.})\\ false \text{ if } \Pi \not\models \phi \end{cases}\\
\delta(\langle\psi?\rangle\varphi,\Pi) &= \delta(\psi,\Pi) \wedge \delta(\varphi,\Pi)\\
\delta(\langle\rho_1+\rho_2\rangle\varphi,\Pi) &= \delta(\langle\rho_1\rangle\varphi,\Pi) \vee \delta(\langle\rho_2\rangle\varphi,\Pi)\\
\delta(\langle\rho_1;\rho_2\rangle\varphi,\Pi) &= \delta(\langle\rho_1\rangle\langle\rho_2\rangle\varphi,\Pi)\\
\delta(\langle\rho^*\rangle\varphi,\Pi) &= \delta(\varphi,\Pi) \vee \delta(\langle\rho\rangle F_{\langle\rho^*\rangle\varphi},\Pi)\\
\delta([\phi]\varphi,\Pi) &= \begin{cases} E(\varphi) \text{ if } \Pi \models \phi \quad (\phi \text{ prop.})\\ true \text{ if } \Pi \not\models \phi \end{cases}\\
\delta([\psi?]\varphi,\Pi) &= \delta(nnf(\neg\psi),\Pi) \vee \delta(\varphi,\Pi)\\
\delta([\rho_1+\rho_2]\varphi,\Pi) &= \delta([\rho_1]\varphi,\Pi) \wedge \delta([\rho_2]\varphi,\Pi)\\
\delta([\rho_1;\rho_2]\varphi,\Pi) &= \delta([\rho_1][\rho_2]\varphi,\Pi)\\
\delta([\rho^*]\varphi,\Pi) &= \delta(\varphi,\Pi) \wedge \delta([\rho]T_{[\rho^*]\varphi},\Pi)\\
\delta(F_\psi,\Pi) &= false\\
\delta(T_\psi,\Pi) &= true
\end{aligned}$$

Fig. 1: Definition of δ, where $E(\varphi)$ recursively replaces in φ all occurrences of atoms of the form T_ψ and F_ψ by ψ.

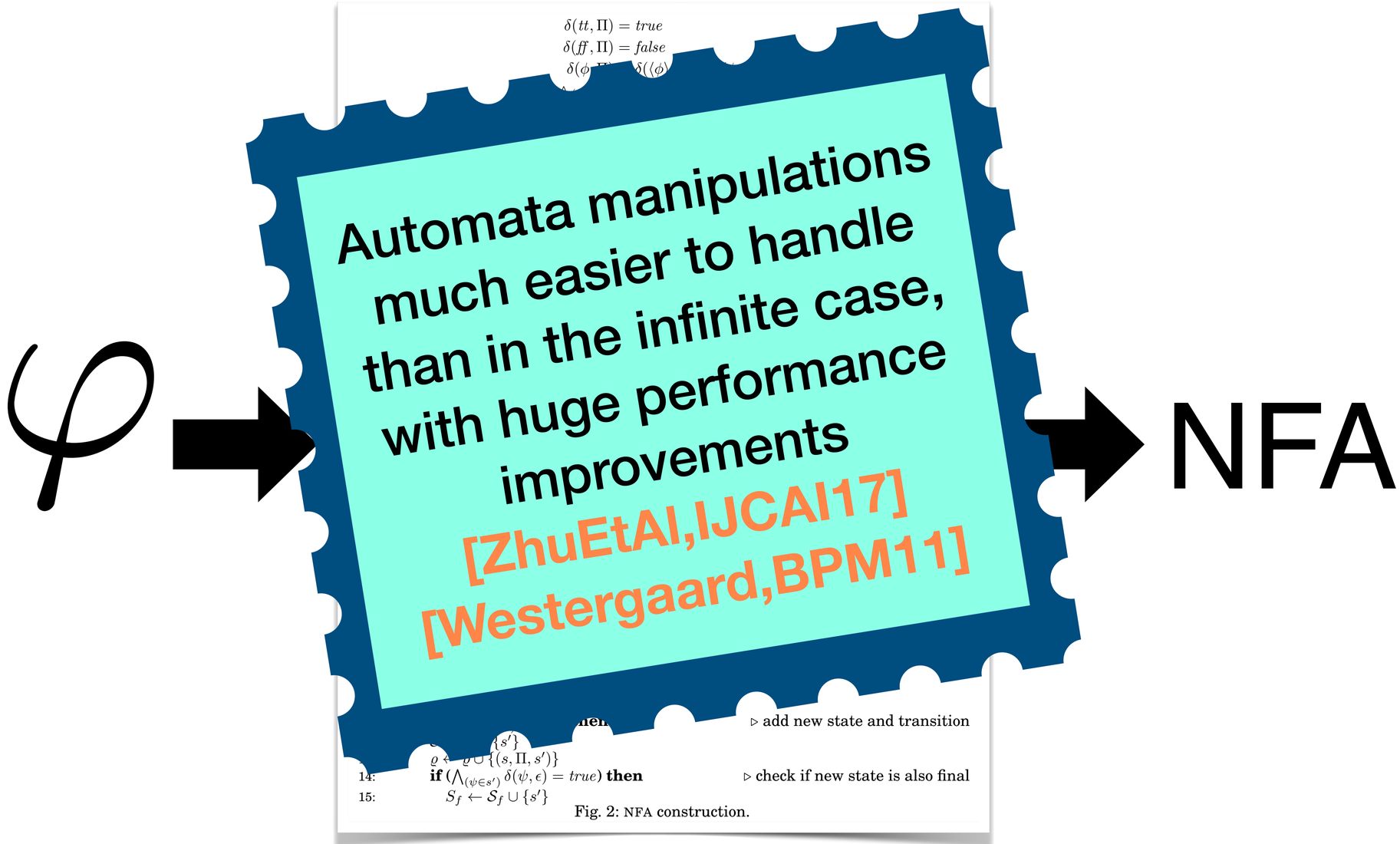
algorithm LDL$_f$2NFA
input LDL$_f$ formula φ
output NFA $\mathcal{A}(\varphi) = (2^{\mathcal{P}}, \mathcal{S}, s_0, \varrho, S_f)$
$s_0 \leftarrow \{\varphi\}$ ▷ set the initial state
$S_f \leftarrow \{\emptyset\}$ ▷ set final states
if $(\delta(\varphi,\epsilon) = true)$ **then** ▷ check if initial state is also final
  $\mathcal{S}_f \leftarrow \mathcal{S}_f \cup \{s_0\}$
$\mathcal{S} \leftarrow \{s_0, \emptyset\}$, $\varrho \leftarrow \emptyset$
while ($\mathcal{S}$ or ϱ change) **do**
  for $(s \in \mathcal{S})$ **do**
    if $(s' \models \bigwedge_{(\psi \in s)} \delta(\psi,\Pi)$ **then** ▷ add new state and transition
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{s'\}$
      $\varrho \leftarrow \varrho \cup \{(s,\Pi,s')\}$
      if $(\bigwedge_{(\psi \in s')} \delta(\psi,\epsilon) = true)$ **then** ▷ check if new state is also final
        $S_f \leftarrow \mathcal{S}_f \cup \{s'\}$

Fig. 2: NFA construction.

$$\rightarrow \text{NFA}$$

Few lines of code

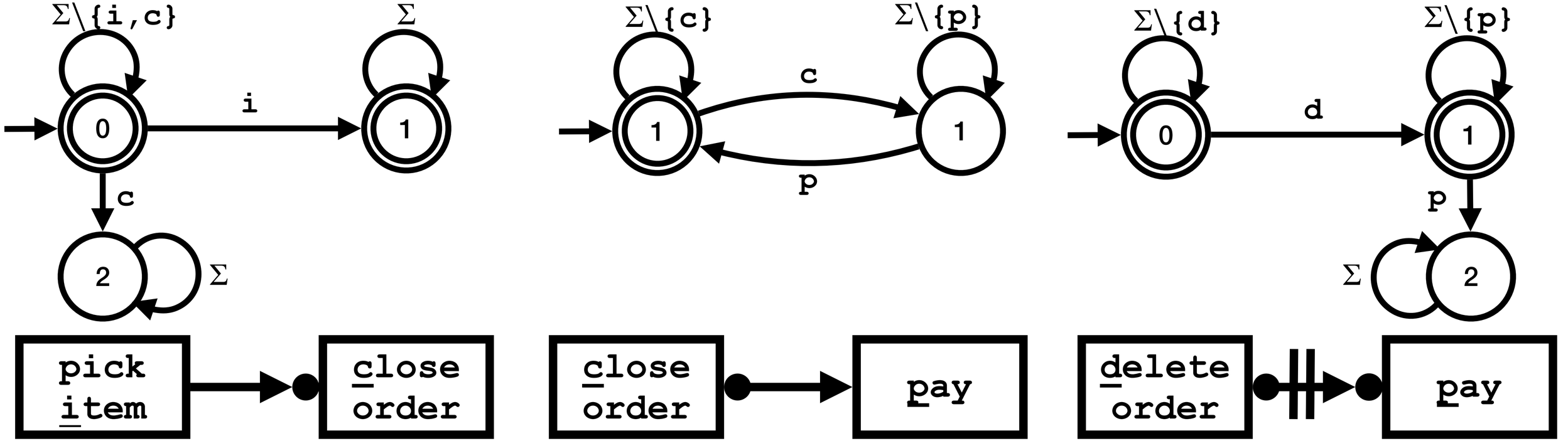
[____,TOSEM2022]



Constraint automata

Template: pre-compiled into a DFA

Constraint: grounds the template DFA on specific activities



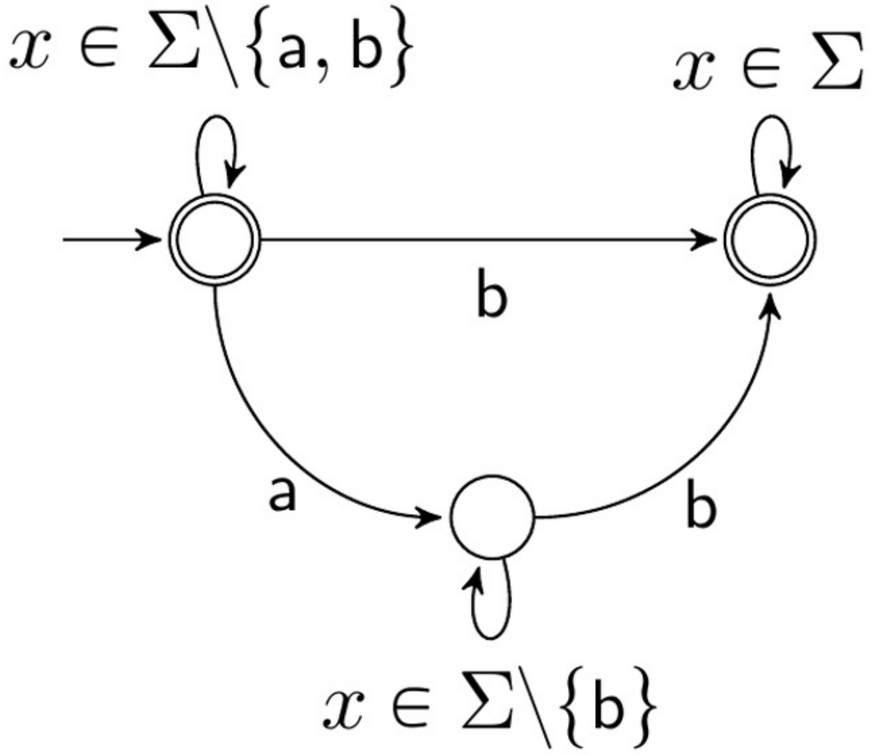
Combining constraints

responded existence(a,b)

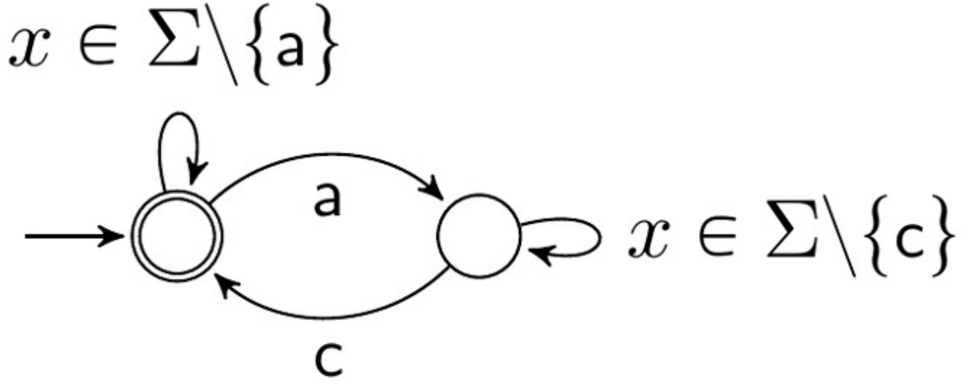
response(a,c)

Combining constraints

responded existence(a,b)

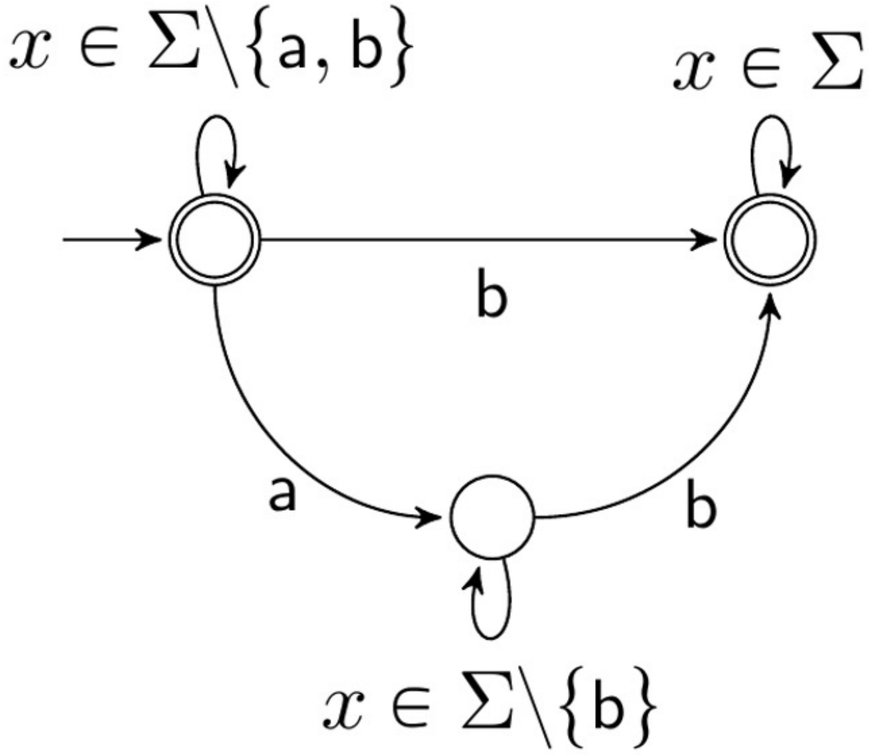


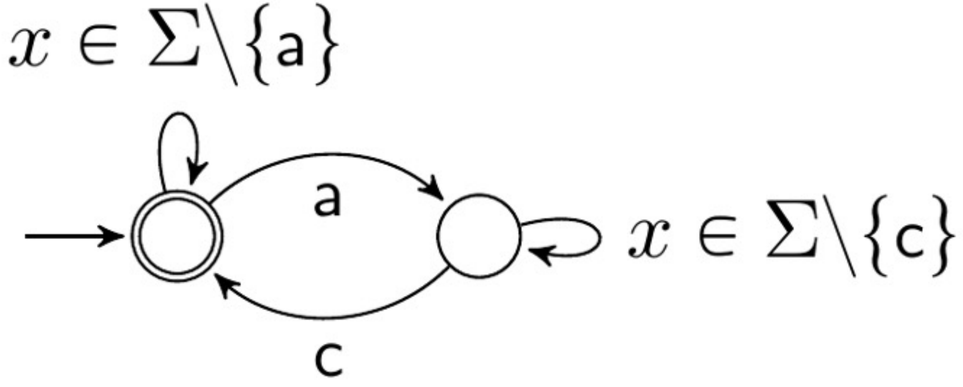
response(a,c)



Combining constraints

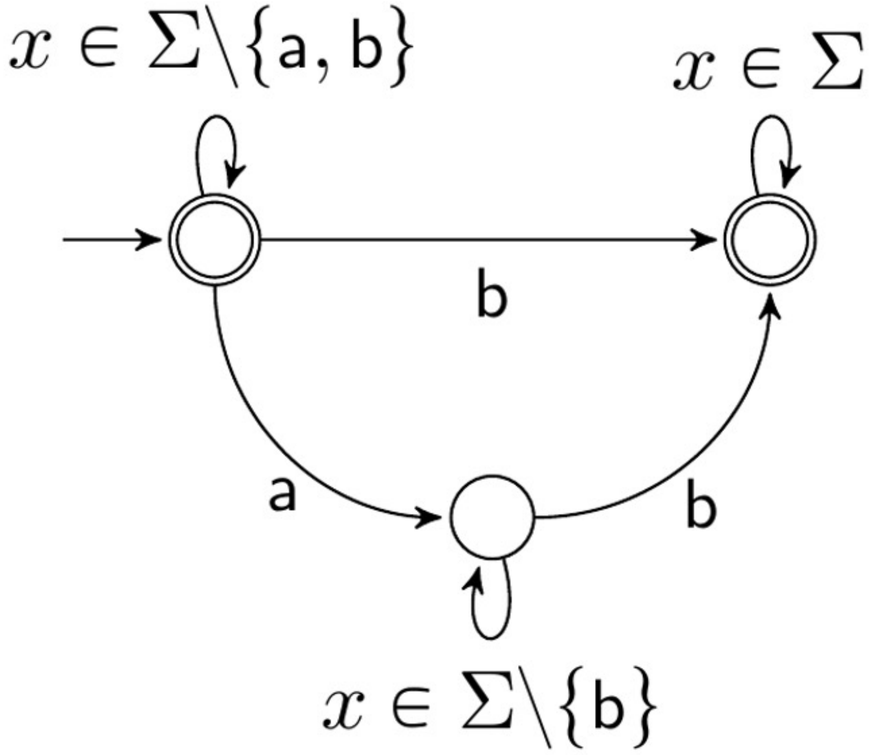
responded existence(a,b)	AND	response(a,c)



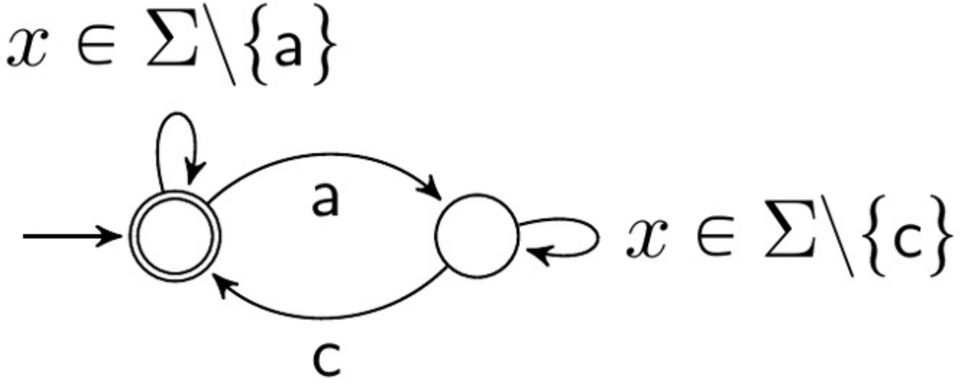


Combining constraints

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

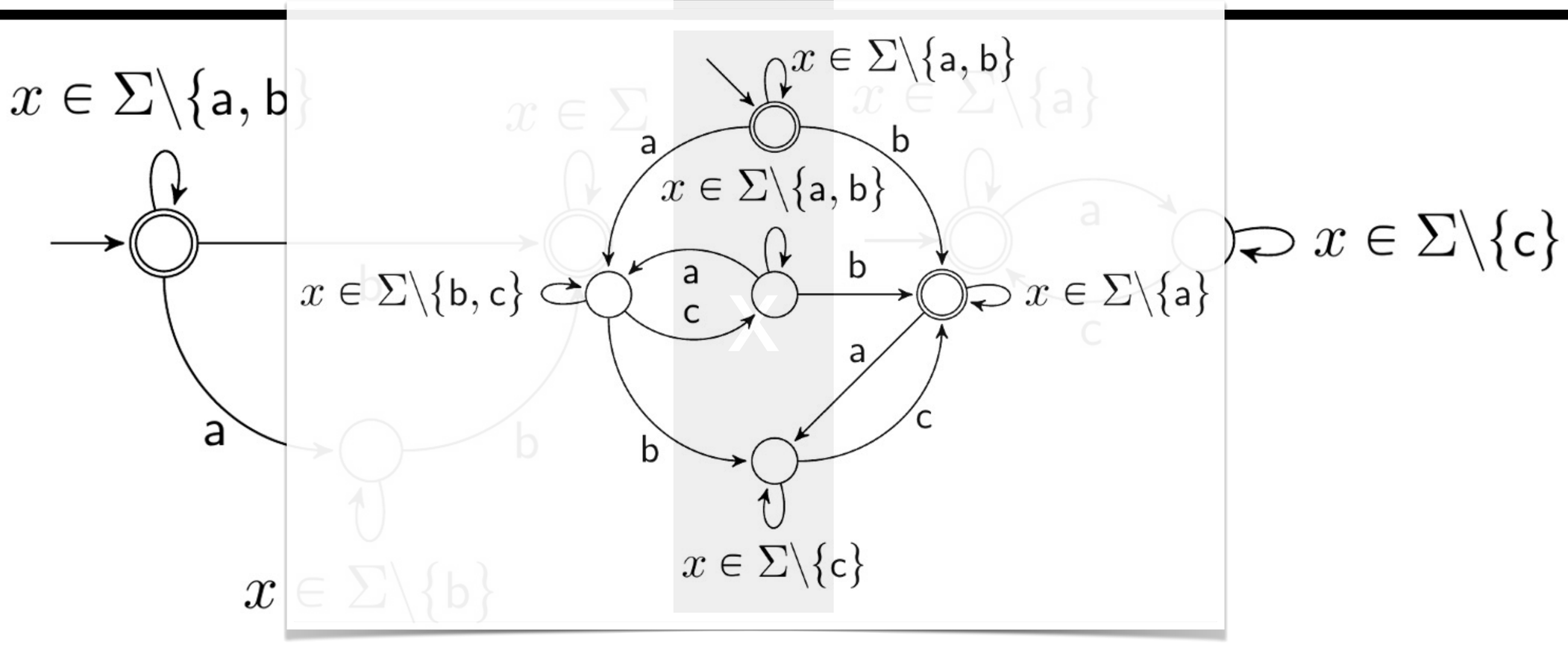


X



Combining constraints

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



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**

Constraints: hard or soft?

Logically: **hard**

Conceptually: **not so clear**

- Model level: mix of constraints of **different nature**
  - Physical, best practices, policies, legal, … **[AdamoEtAl,InfSys2021]**
- Hard at the IS level <-> hard or soft in reality
  - **Manual task** vs **user-interaction task**
  - **Ontological reversal** **[BaskervilleEtAl,MISQ2020]**

Constraints: hard or soft?

Logically: **hard**

Conceptually: **not so clear**

- Model level: mix of constraints of **different nature**
  - Physical, best practices, policies, legal, … **[AdamoEtAl,InfSys2021]**
- Hard at the IS level <-> hard or soft in reality
  - **Manual task** vs **user-interaction task**
  - **Ontological reversal** **[BaskervilleEtAl,MISQ2020]**

ABPMS needs to to account for deviations, at runtime

Monitoring and operational support

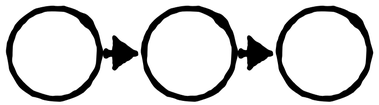
monitoring

Track a **running process execution** to **check conformance** to a reference process model

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

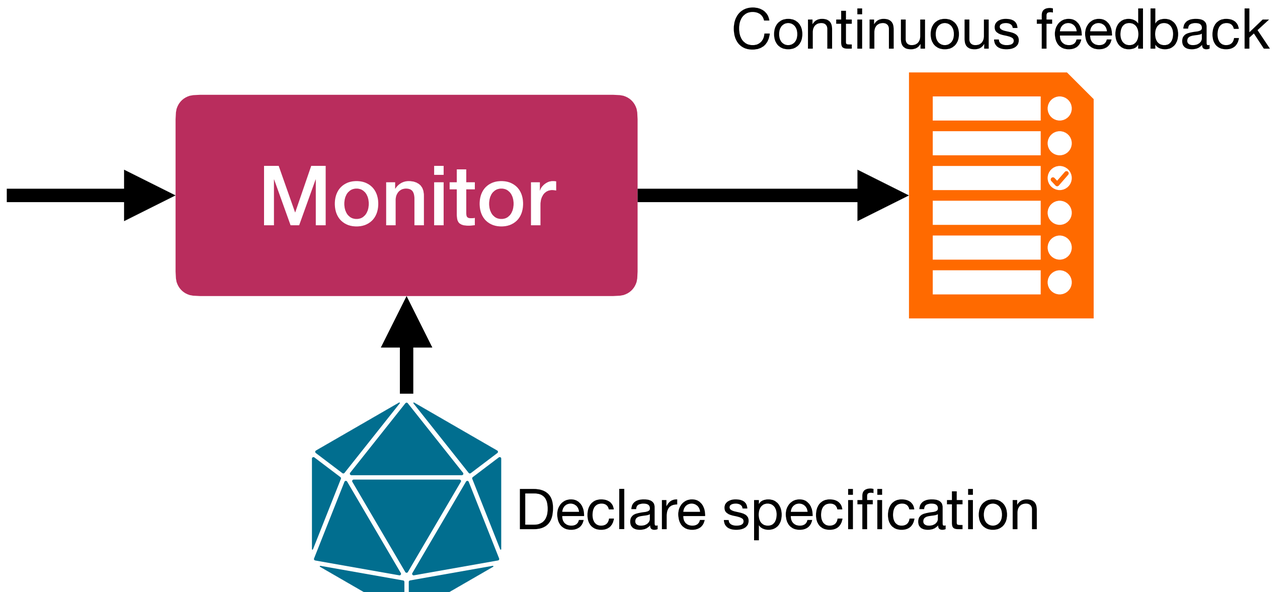
One of the main **operational support tasks**

- Complementary to **predictive monitoring!**



t

Evolving trace



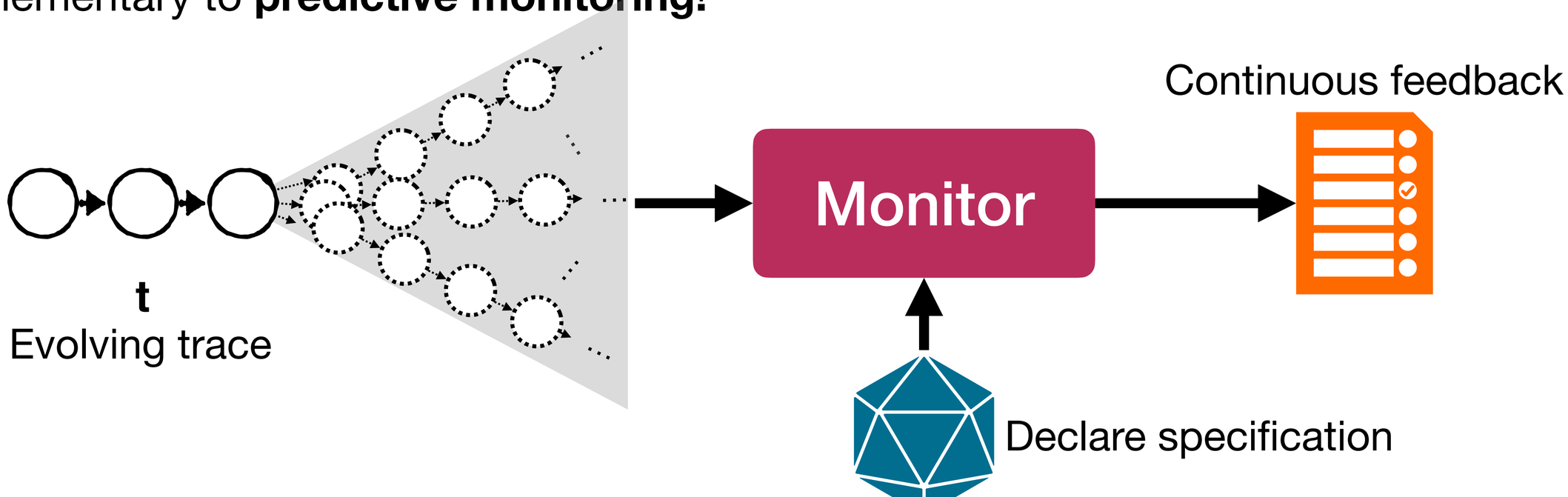
(Anticipatory) monitoring

Track a **running process execution** to **check conformance** to a reference process model

- Goal: Detect and report **fine-grained feedback** and **deviations** **as early as possible**

One of the main **operational support tasks**

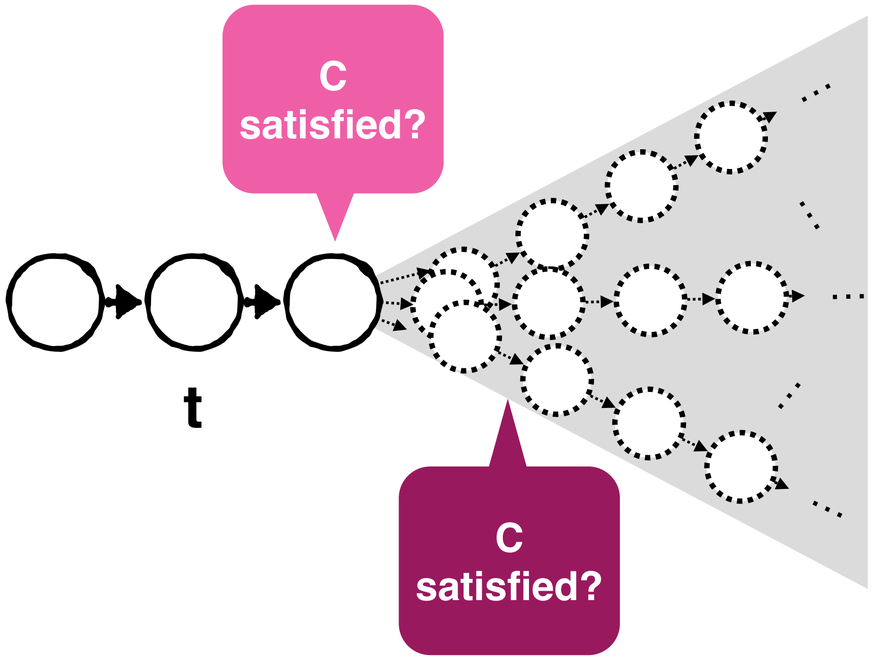
- Complementary to **predictive monitoring!**



Fine-grained feedback

Refined analysis of the “truth value” of a constraint, looking into (all) **possible futures**

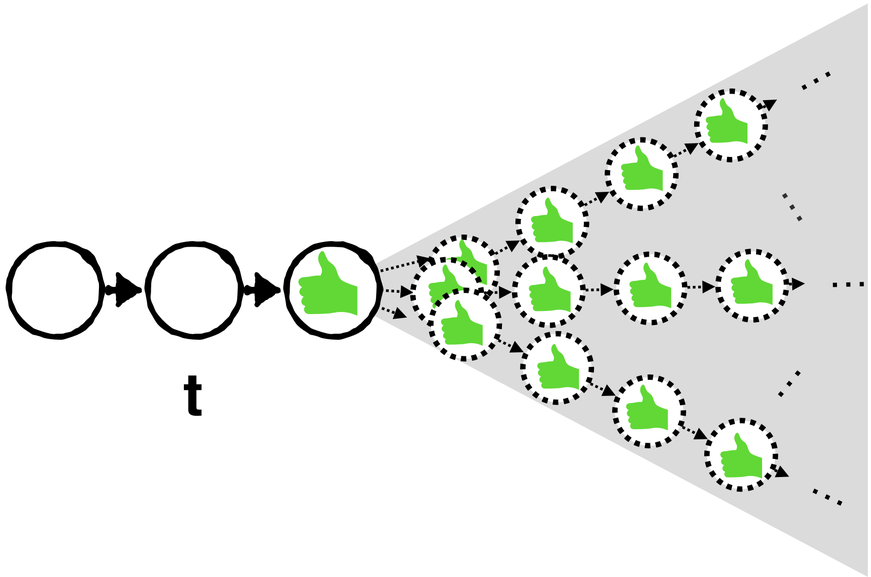
Consider a partial trace **t**, and a constraint **C**...



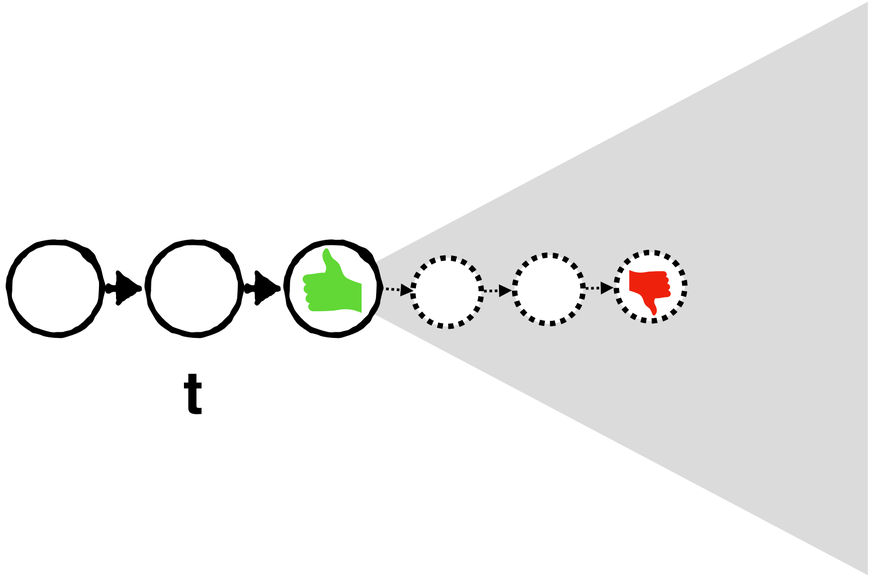
RV-LTL truth values

[BauerEtAl,InfCom2010]

C is **permanently satisfied** if **t** satisfies **C** and no matter how **t** is extended, **C** will stay satisfied



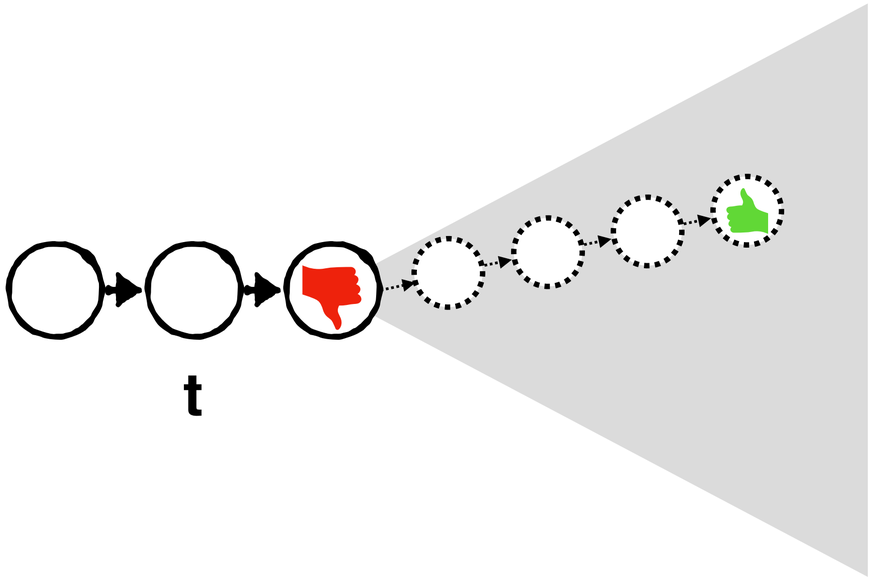
C is **currently satisfied** if **t** satisfies **C** but there is a continuation of **t** that violates **C**



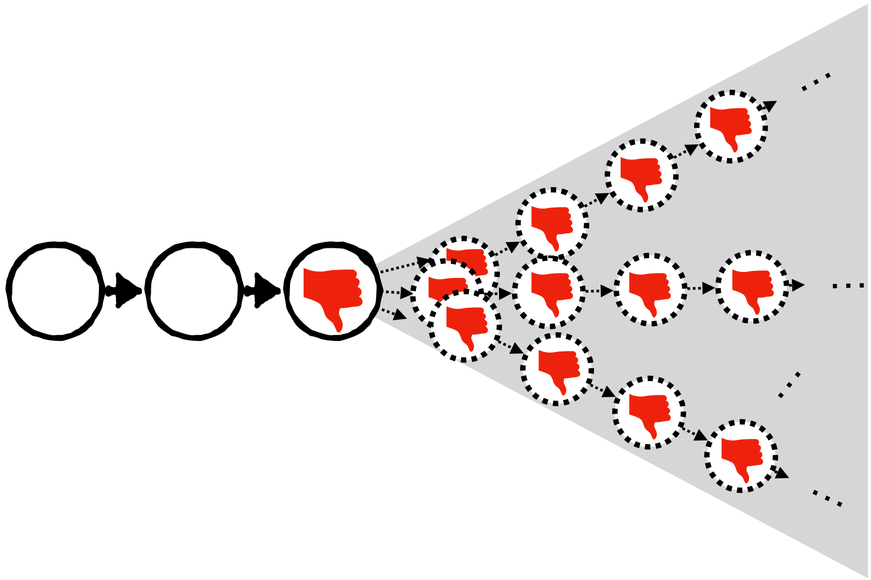
RV-LTL truth values

[BauerEtAl,InfCom2010]

C is **currently violated** if **t** violates **C** but there is a continuation that leads to satisfy **C**



C is **permanently violated** if **t** violates **C** and no matter how **t** is extended, C will stay violated



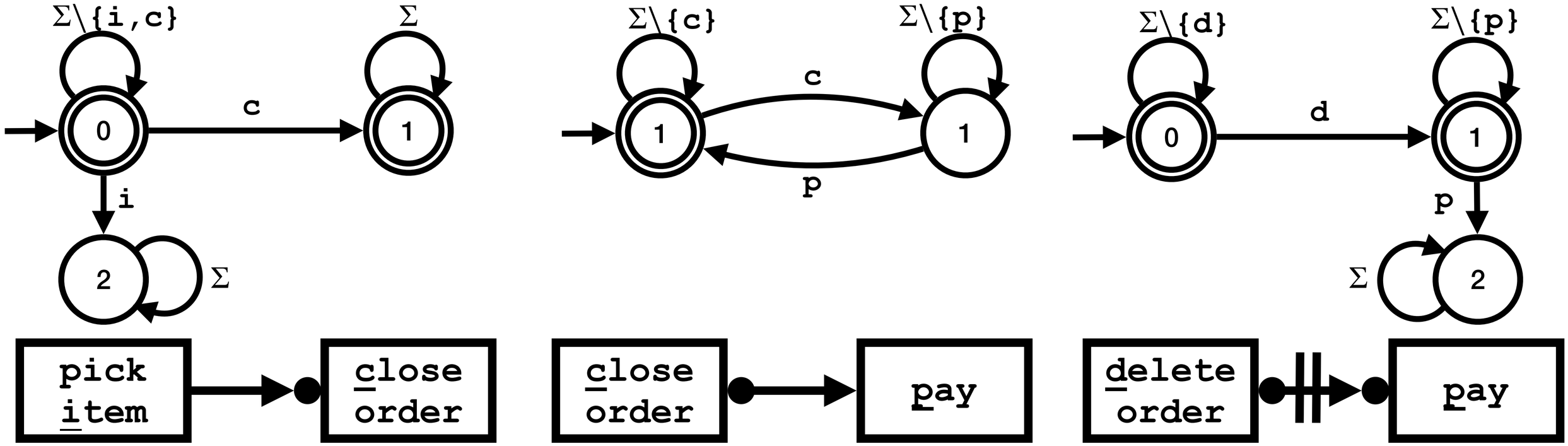
RV-LTL on finite traces

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

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

- Again standard DFAs -> **all formulae of LTLf are monitorable**

Each **state** of the DFA: **colored** with an **RV-LTL value** via simple reachability checks



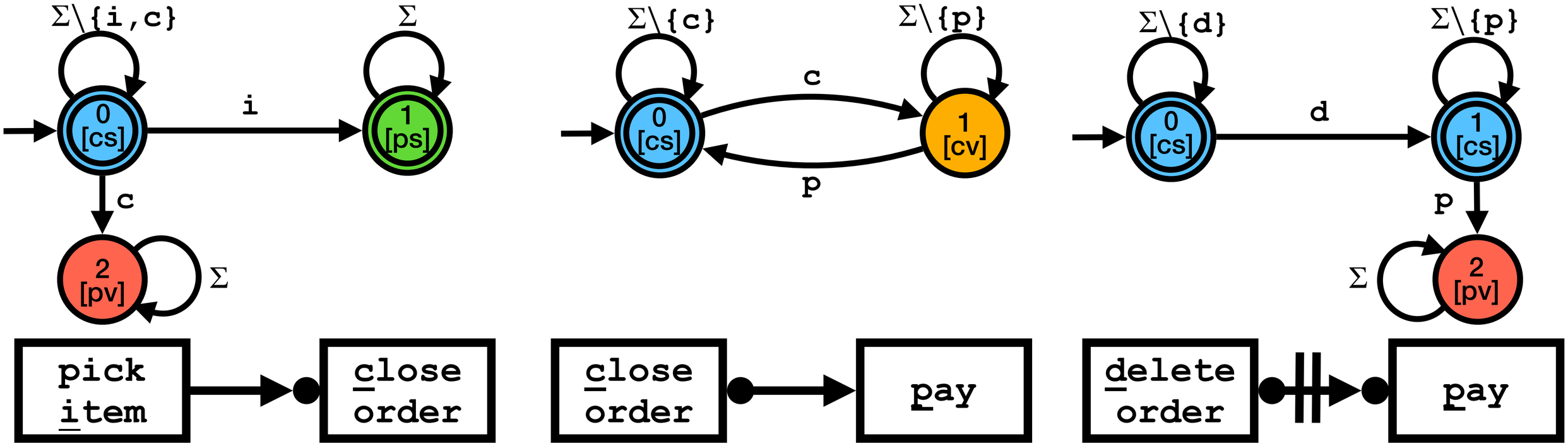
RV-LTL on finite traces

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

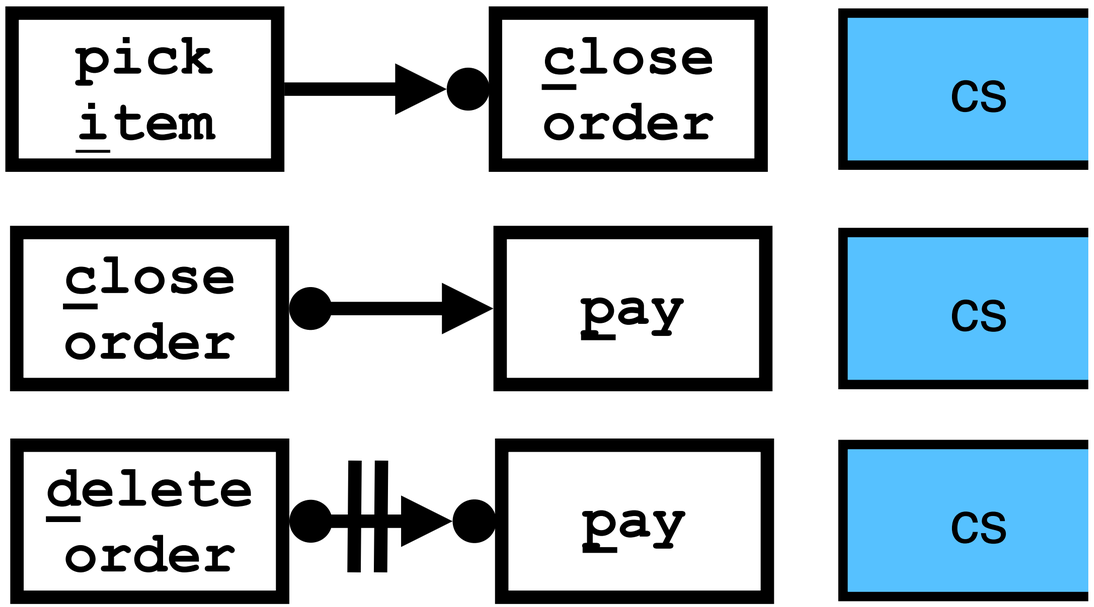
Suffixes of the current trace: each with **unbounded, finite length**

- Again standard DFAs -> **all formulae of LTLf are monitorable**

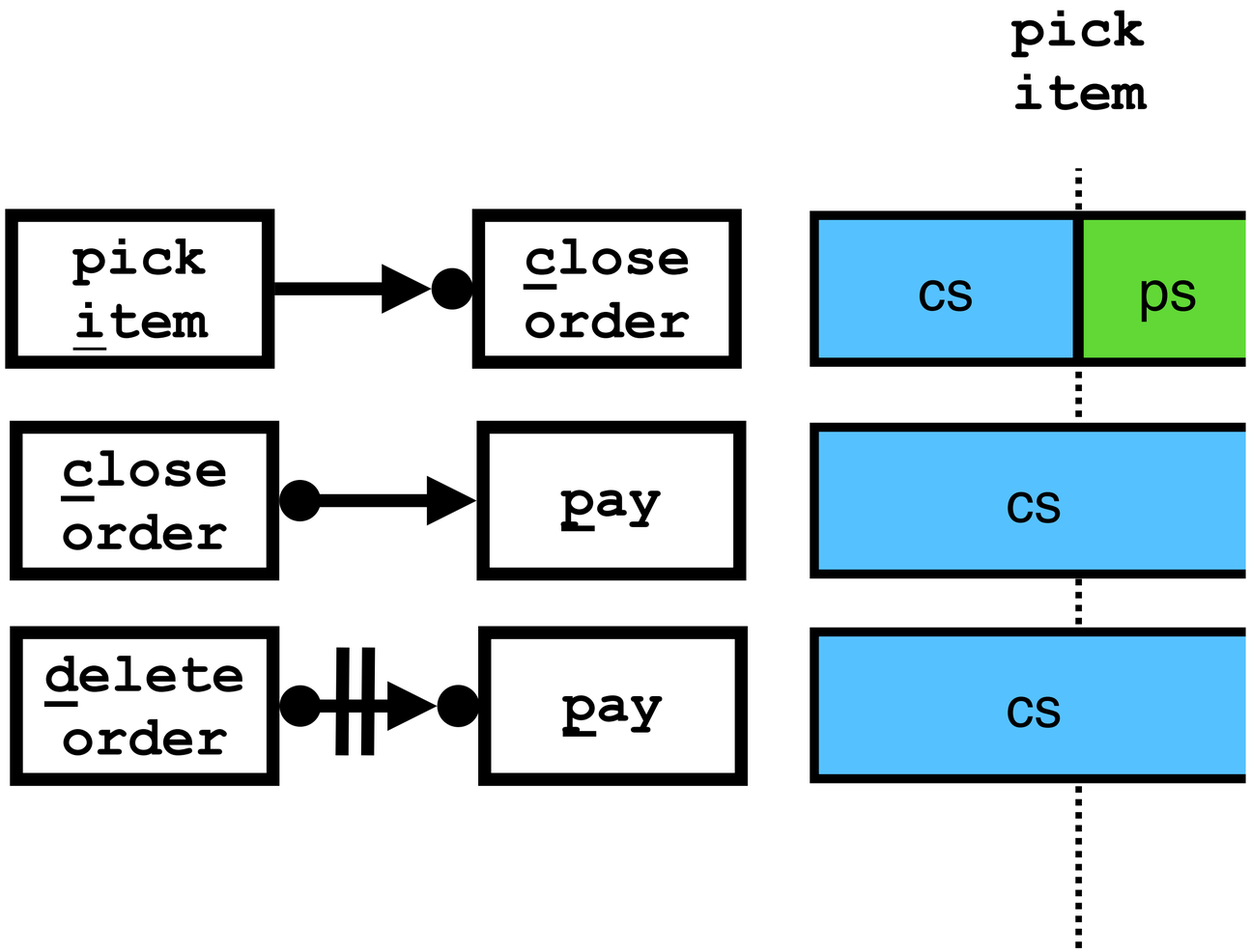
Each **state** of the DFA: **colored** with an **RV-LTL value** via simple reachability checks



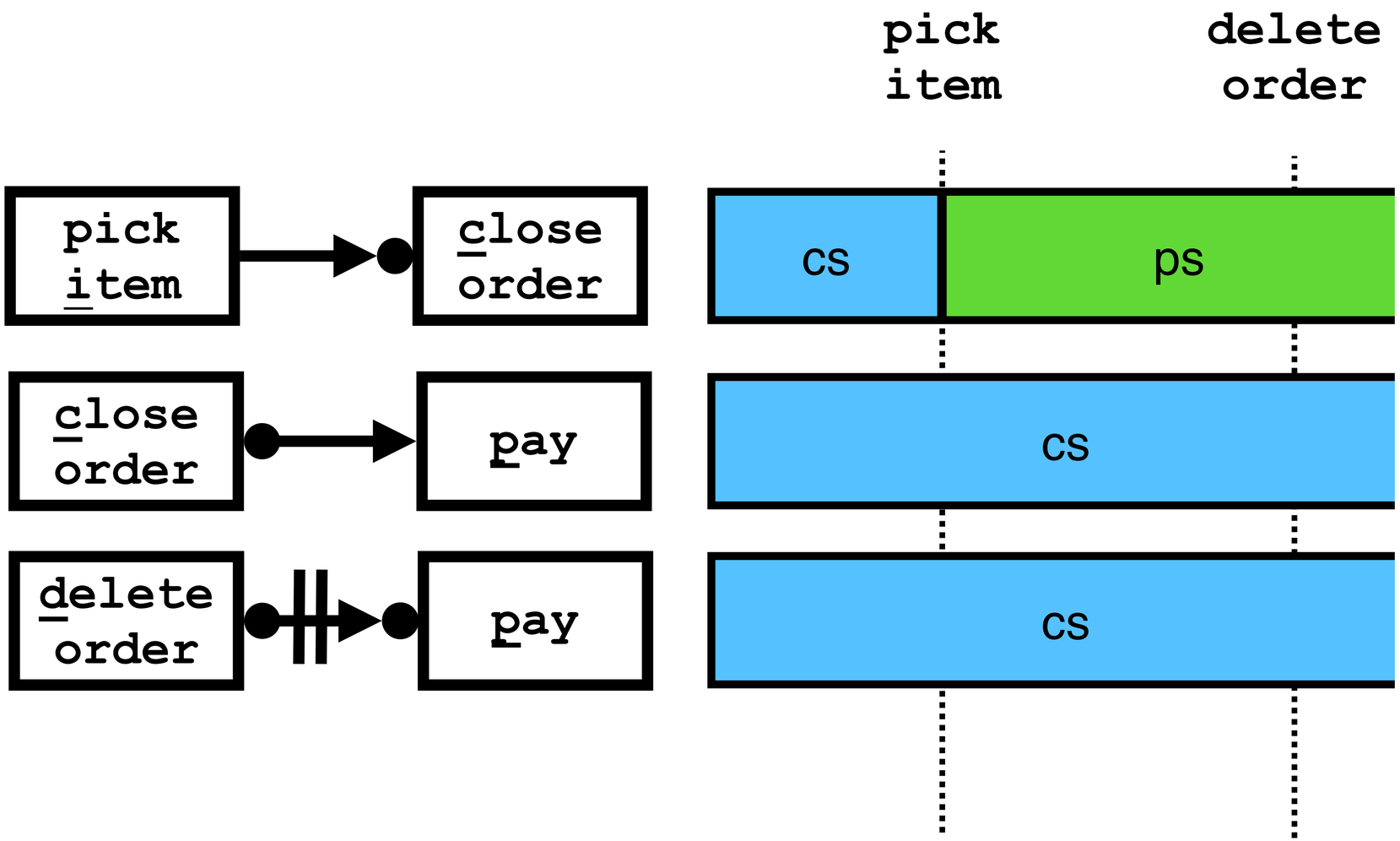
Monitor in action



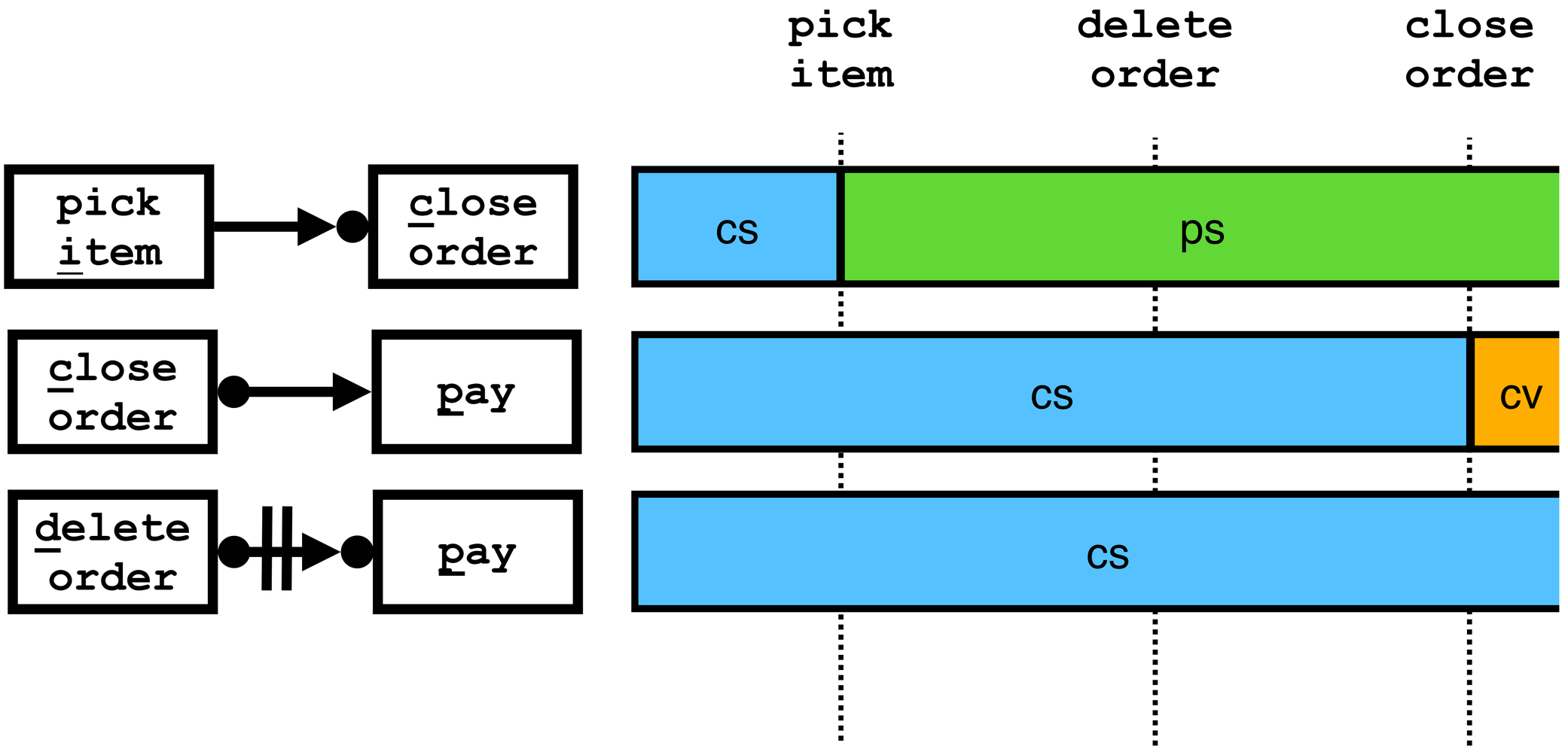
Monitor in action



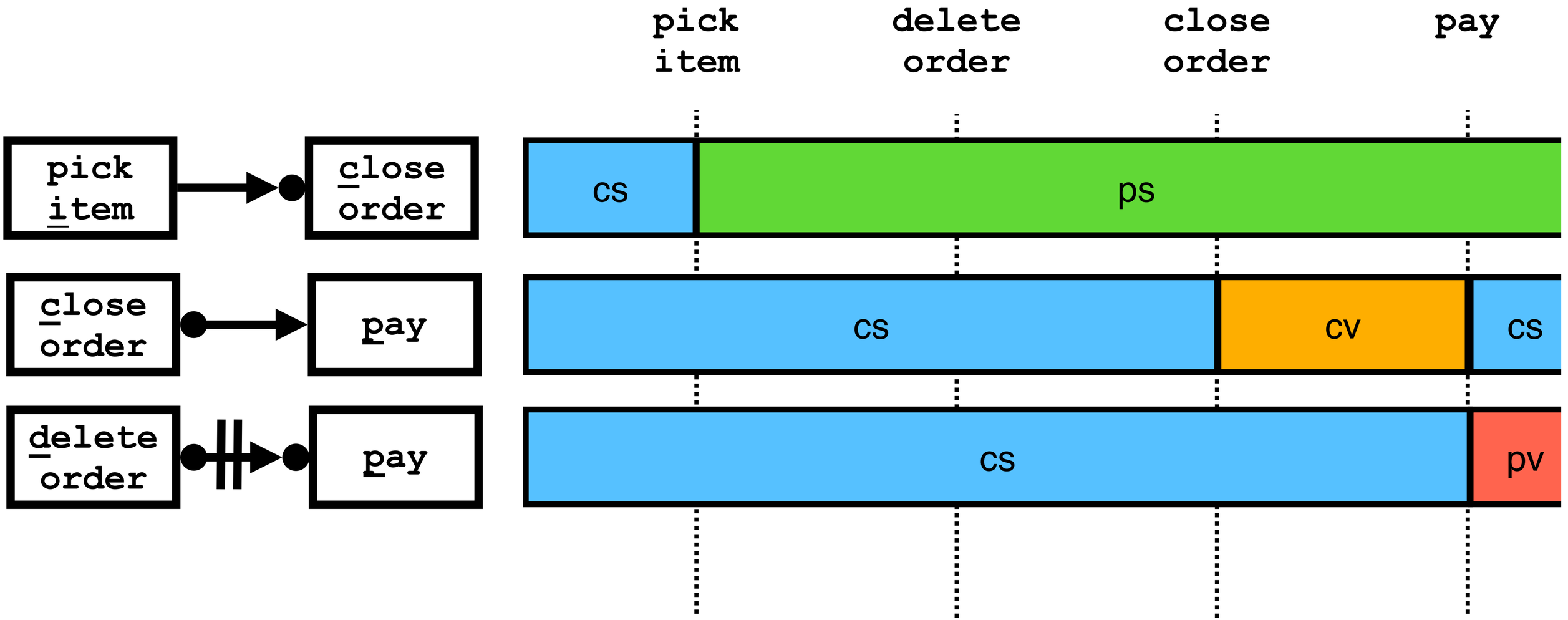
Monitor in action



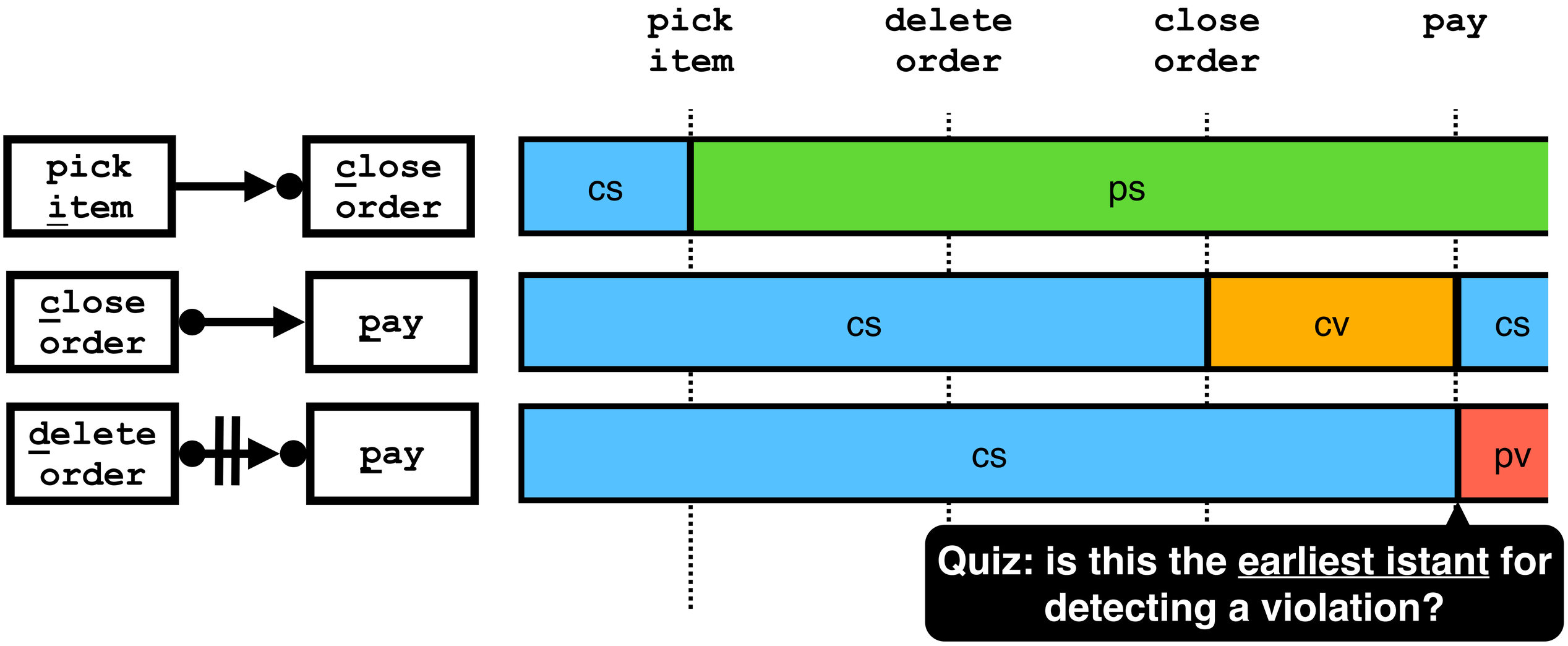
Monitor in action



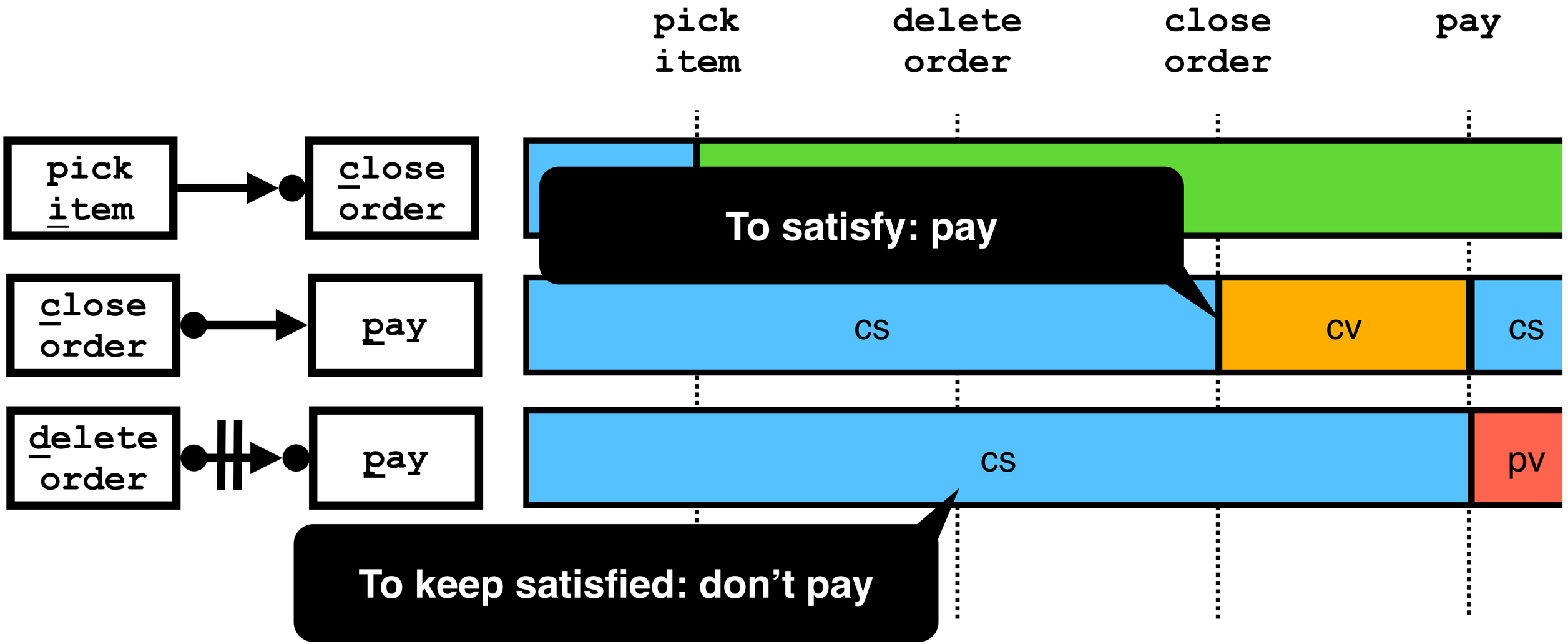
Monitor in action



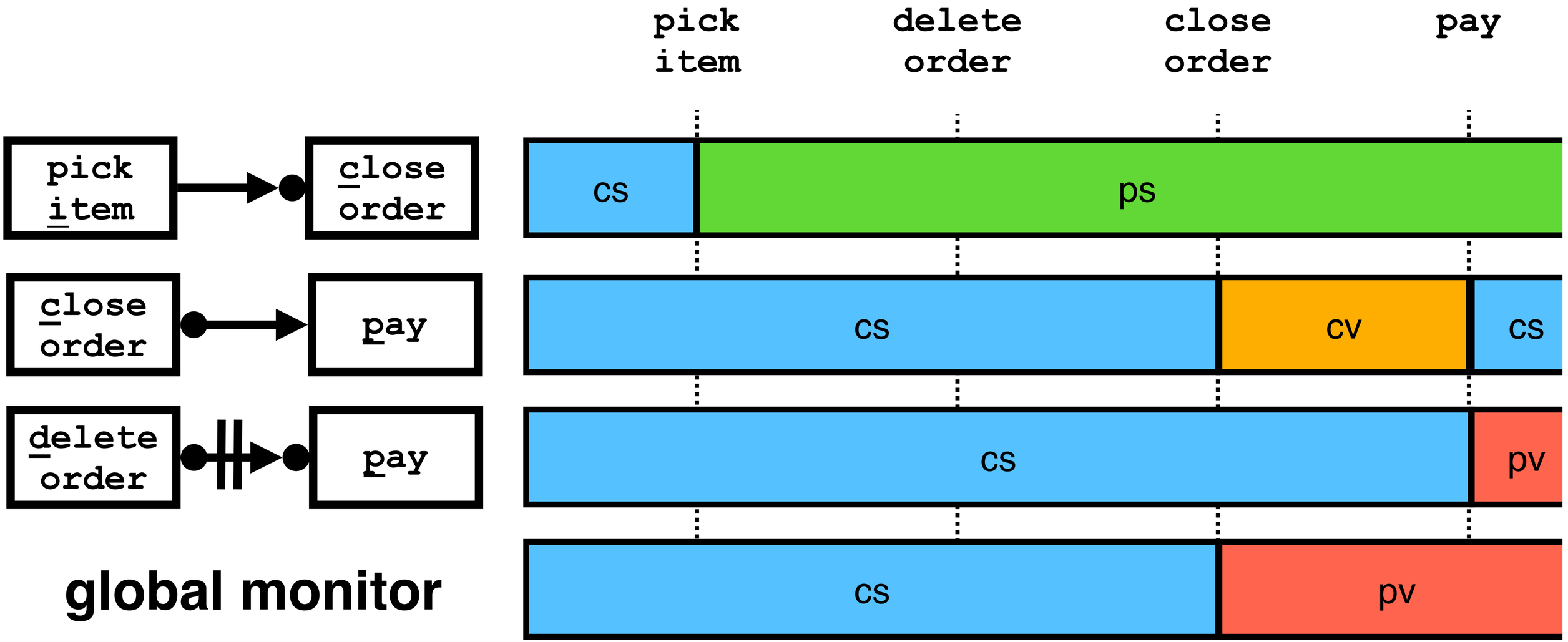
Monitor in action



Monitor in action



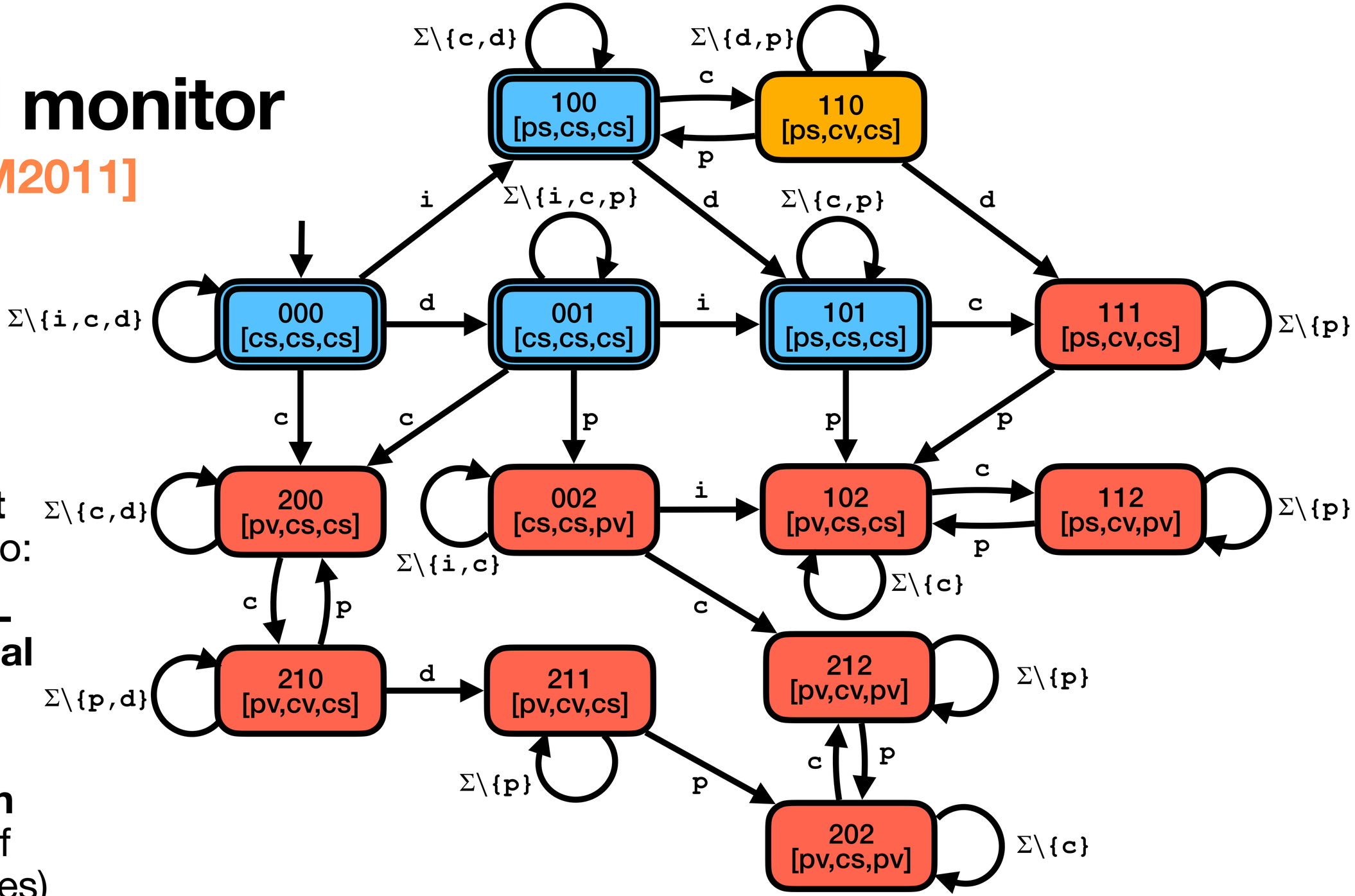
Monitor in action



global monitor

Global monitor

[____,BPM2011]

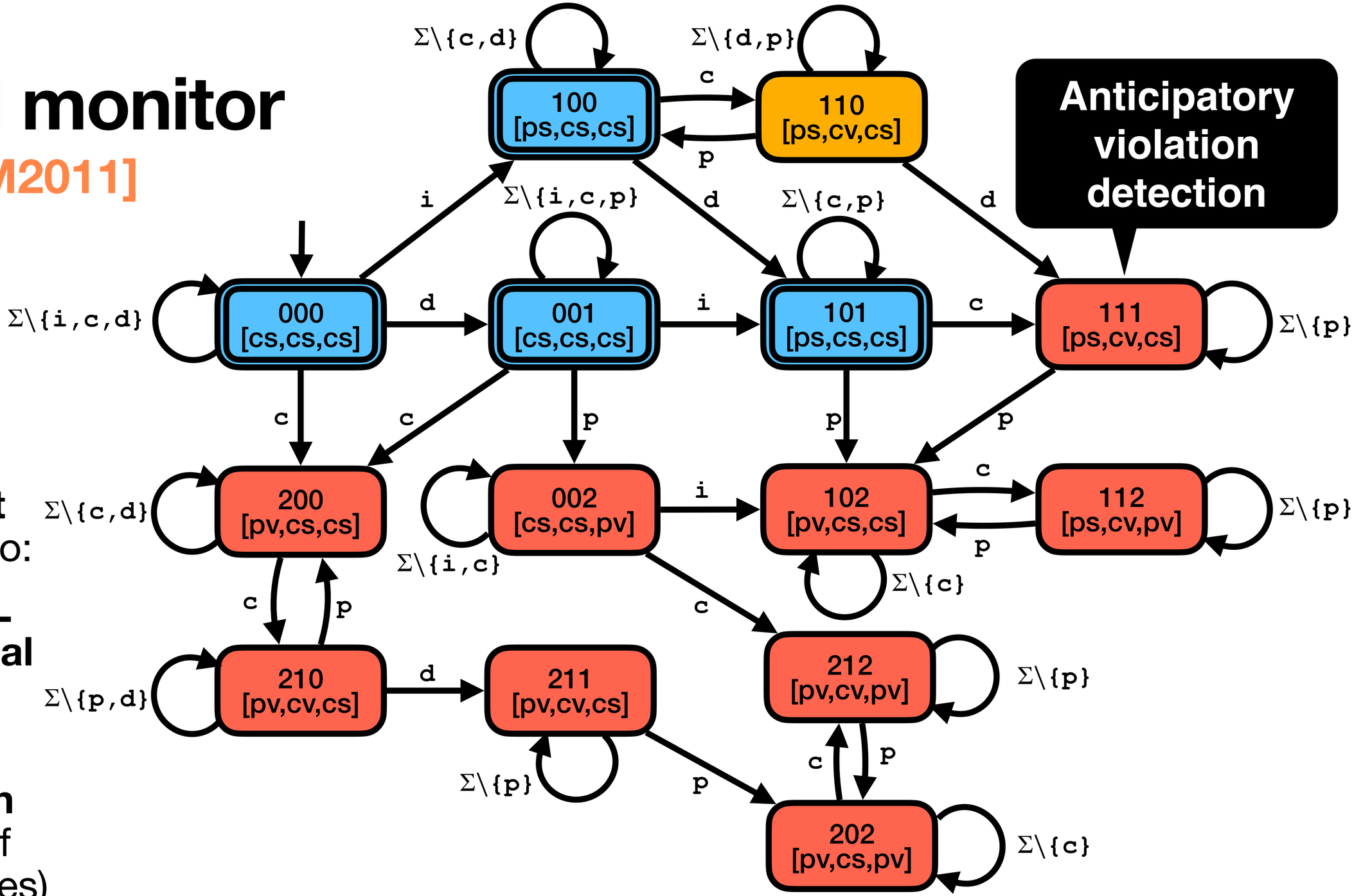


Cross-product with two proviso:

- recall **RV-LTL labels of local constraints**
- **no minimisation** (distinction of violation states)

Global monitor

[____,BPM2011]

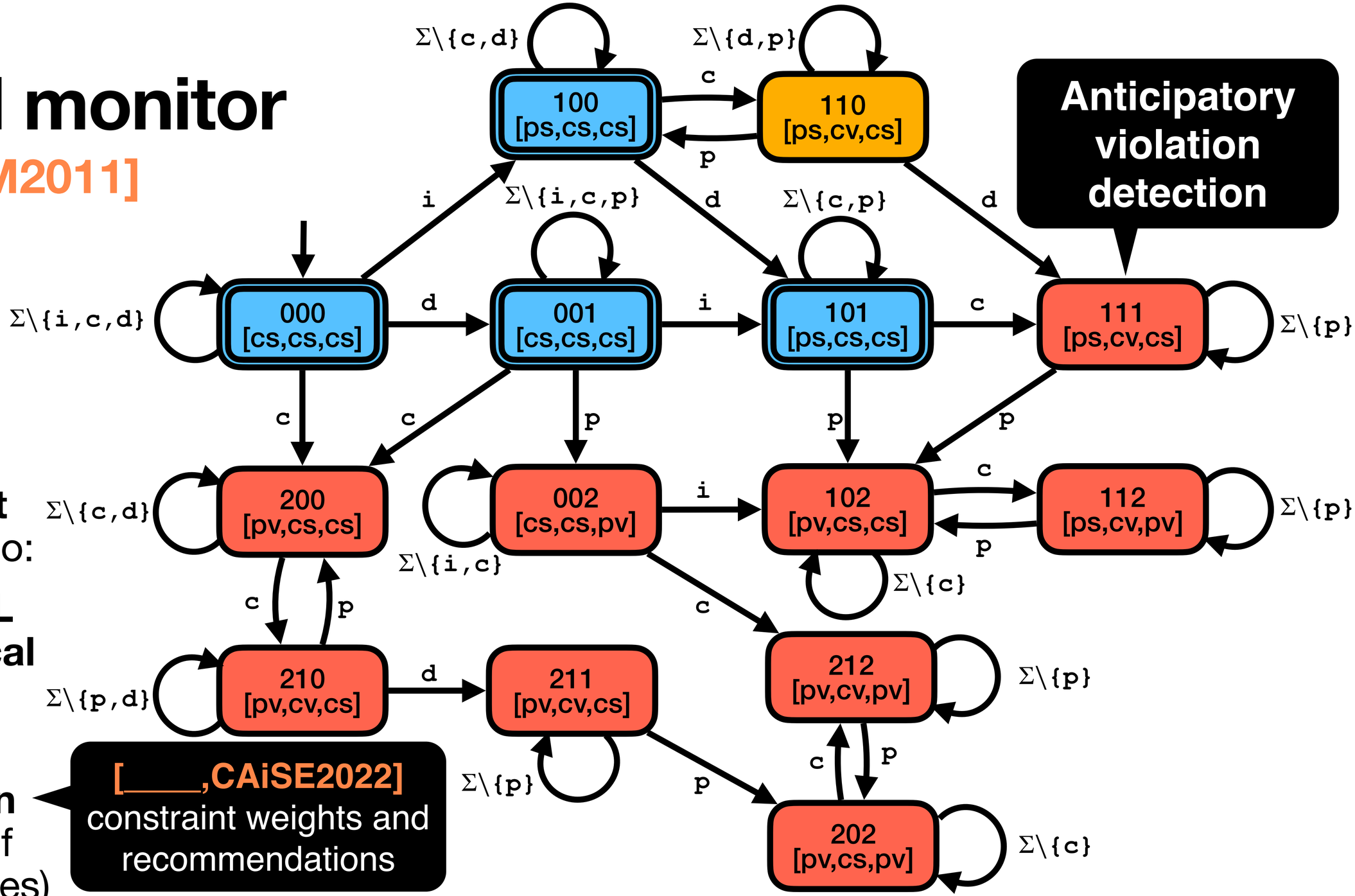


Cross-product with two proviso:

- recall **RV-LTL labels of local constraints**
- **no minimisation** (distinction of violation states)

Global monitor

[____,BPM2011]



Cross-product with two proviso:

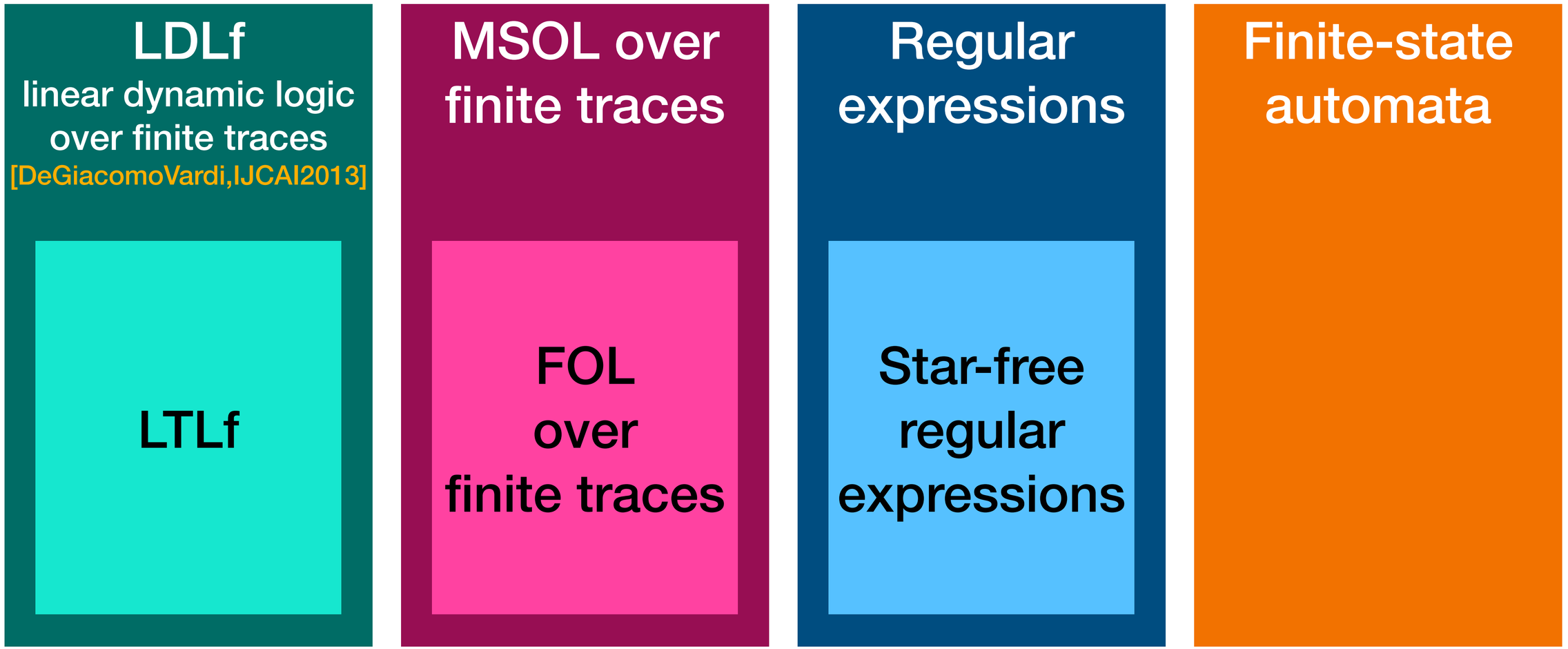
- recall **RV-LTL labels of local constraints**
- **no minimisation** (distinction of violation states)

[____,CAiSE2022] constraint weights and recommendations

Can we do more?

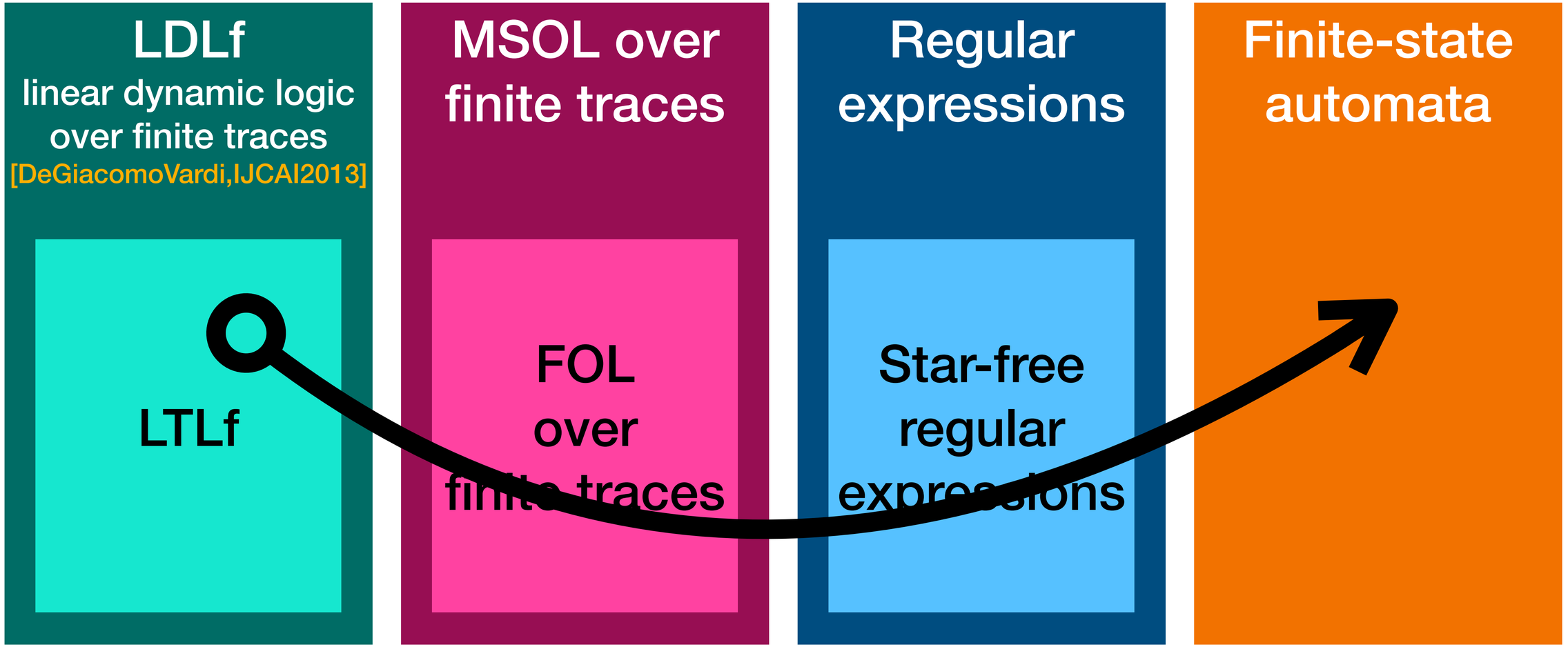
Can we do more?

Can we do more?



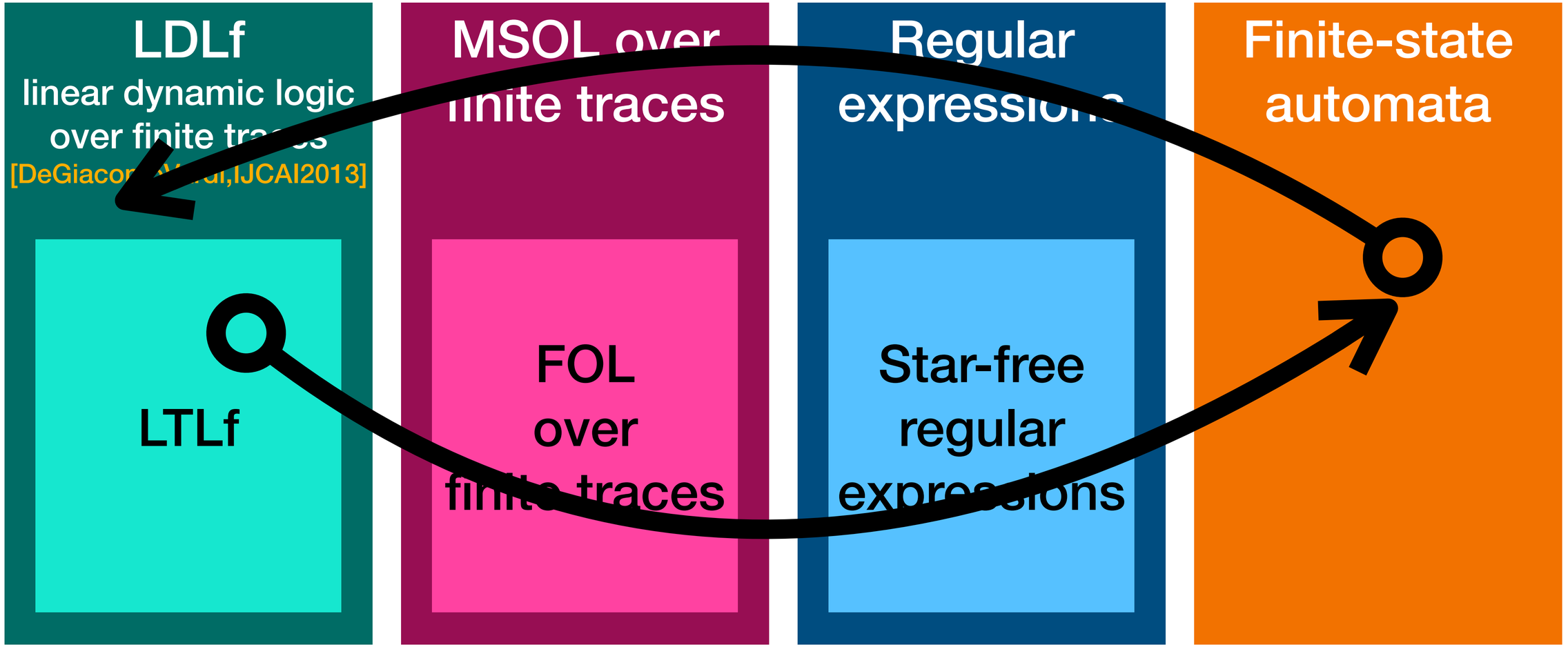
Can we do more?

[____,BPM2014] [____,TOSEM2022]



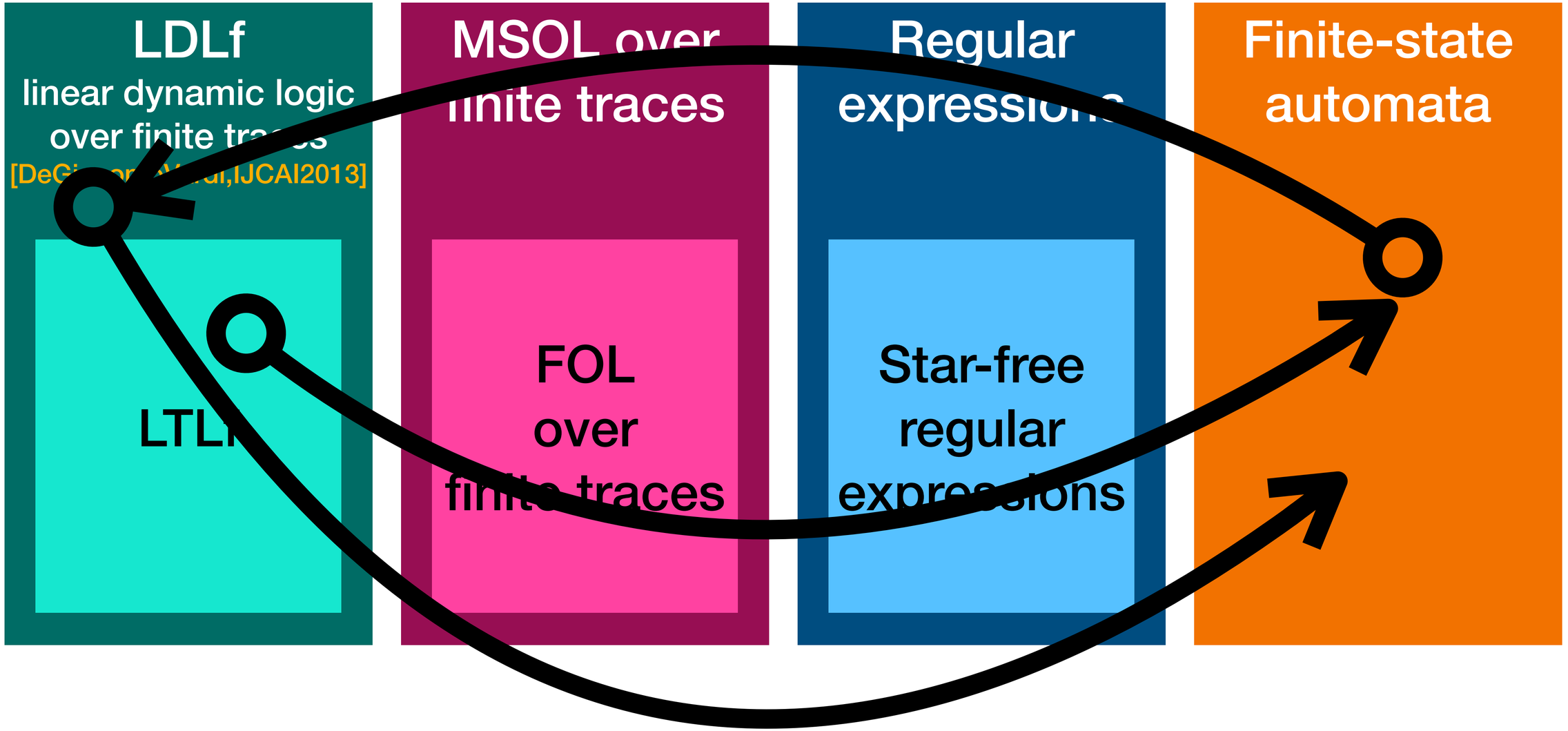
Can we do more?

[____,BPM2014] [____,TOSEM2022]



Can we do more?

[____,BPM2014] [____,TOSEM2022]



From constraints to metaconstraints

[____,BPM2014] [____,TOSEM2022]

LDLf expresses RV-LTL 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**

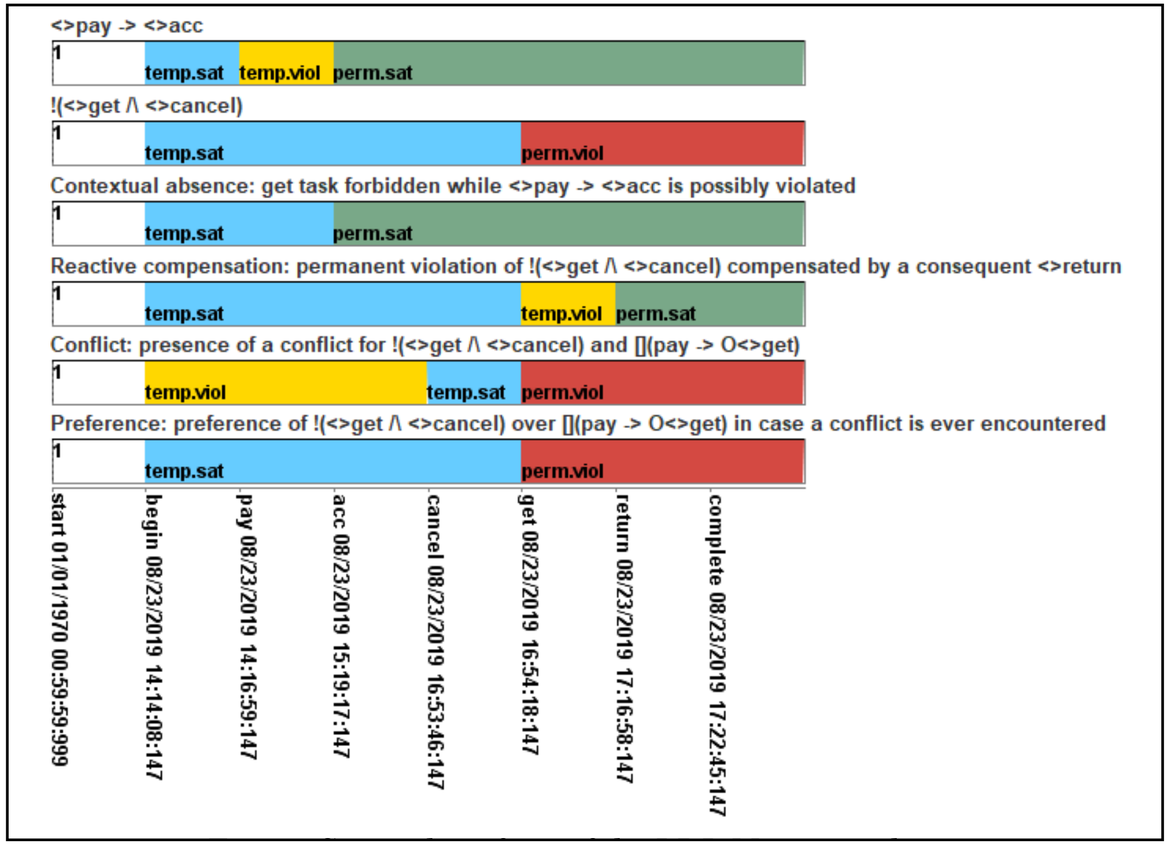
- If **constraint C1** gets **permanently violated**, eventually **satisfy** a **compensation constraint C2**

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

- **[Governatori,EDOC2015]** -> LTL cannot express normative notions
- **[AlechinaEtAl,FLAP2018]**-> not true!

Tooling

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



Adding event attributes and arithmetics

[____,AAAI2022]

Study of **LTLf over numerical variables with arithmetic conditions**

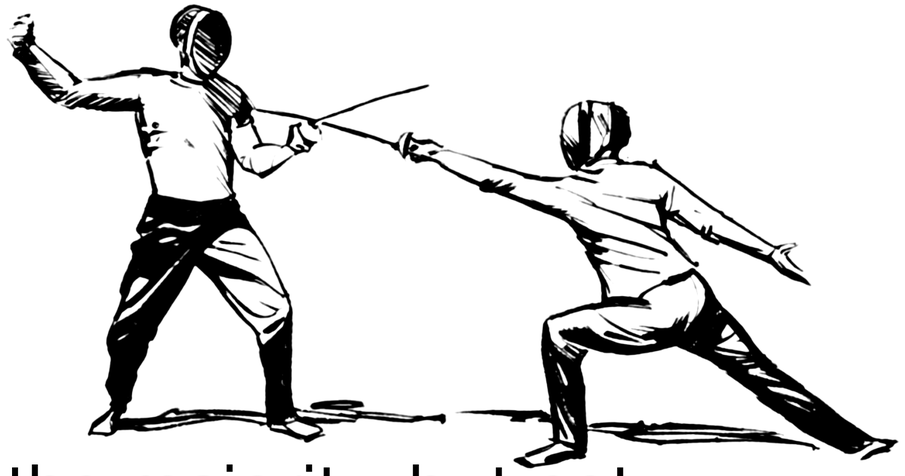
- **Undecidability** around the corner

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

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

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.

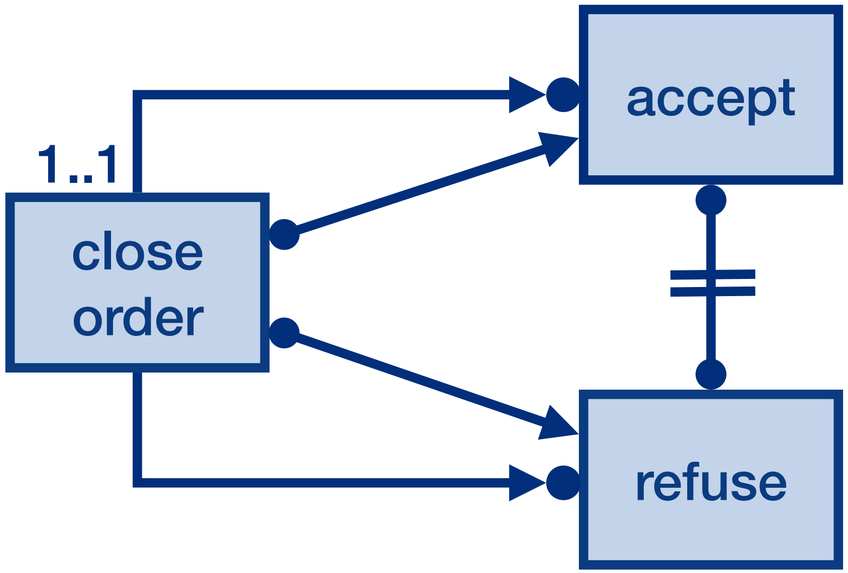
  Only 1% of the orders are canceled after being paid.

- **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.

Dealing with uncertainty

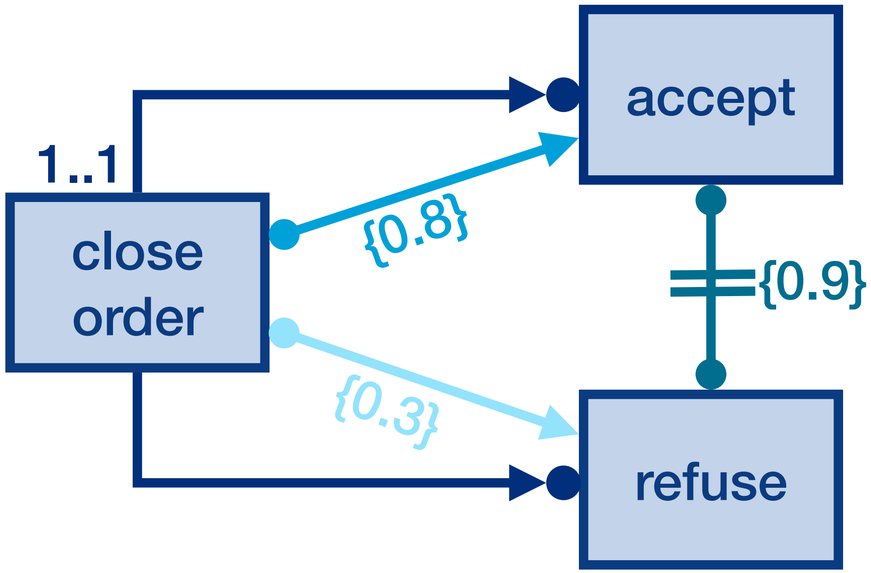
Declare is crisp



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

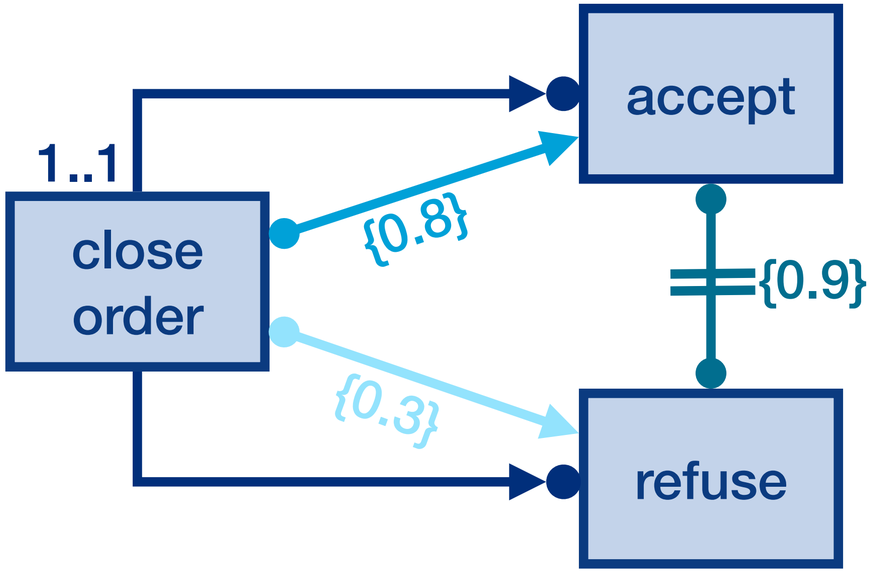
ProbDeclare

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

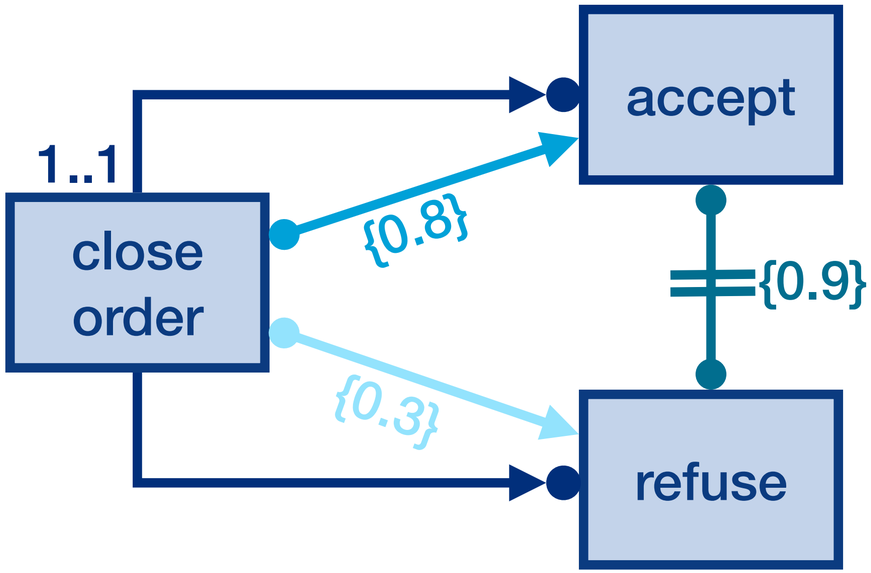
probability operator: $\{ =, \neq, \leq, \geq, <, > \}$

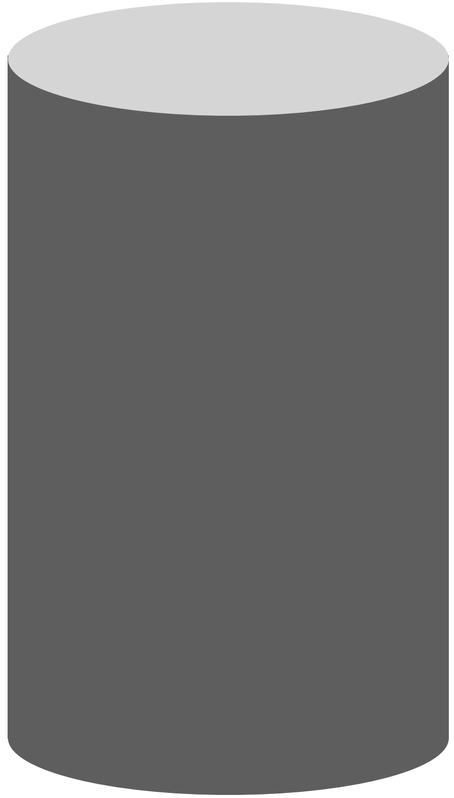
probability reference value: number in [0,1]

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

ProbDeclare

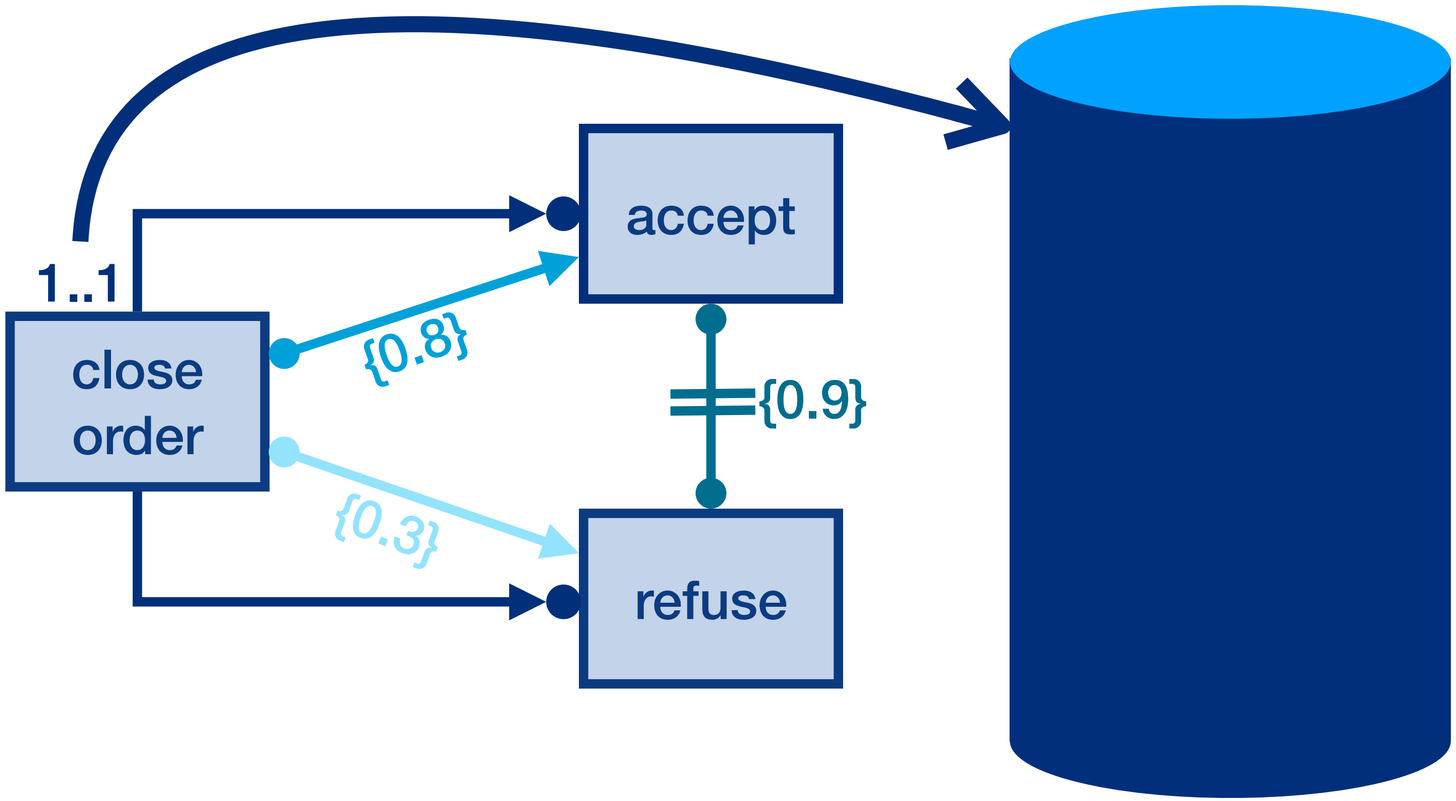
Crisp and uncertain constraints [____,BPM2020] [____,InfSys2022]





ProbDeclare

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

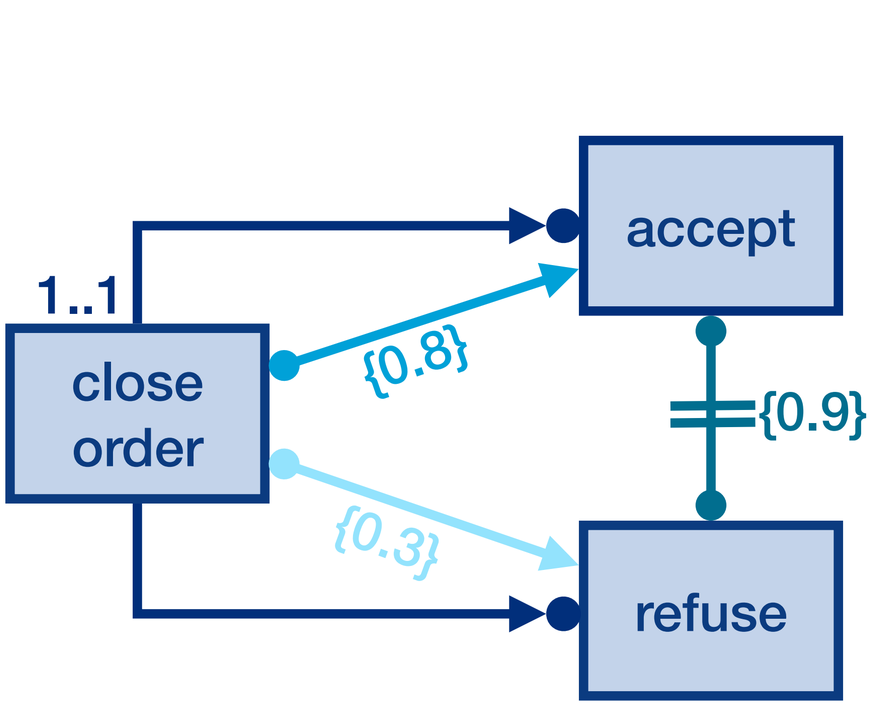


Crisp!

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

ProbDeclare

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

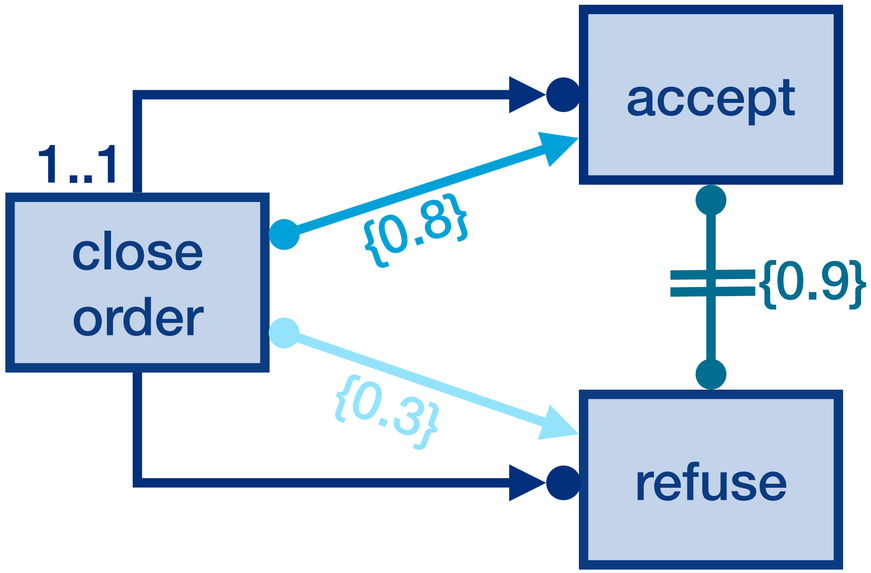


Uncertain!

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 $\rho : \Sigma^* \to [0,1]$ 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 $\tau \vDash \varphi$
- for every **probabilistic constraint** $\langle \varphi, \bowtie, p \rangle$, we have

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

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 $\tau \vDash \varphi$
- for every **probabilistic constraint** $\langle \varphi, \bowtie, p \rangle$, we have

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

Key challenge: again, interplay of constraints

Dealing with “n” probabilistic constraints

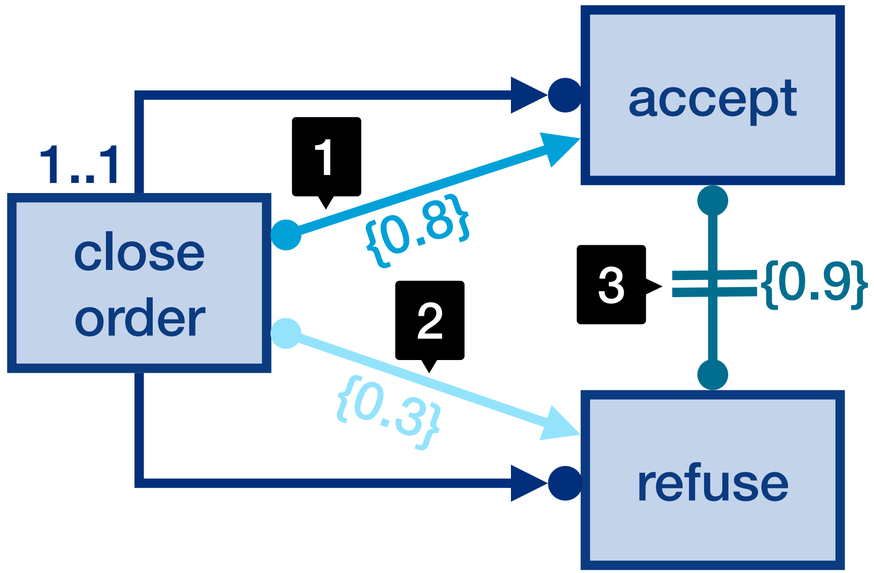
Constraint scenario

Declares which probabilistic constraints must hold, and which are violated

- Constraint violated <-> its negated version holds

Denotes a “process variant”

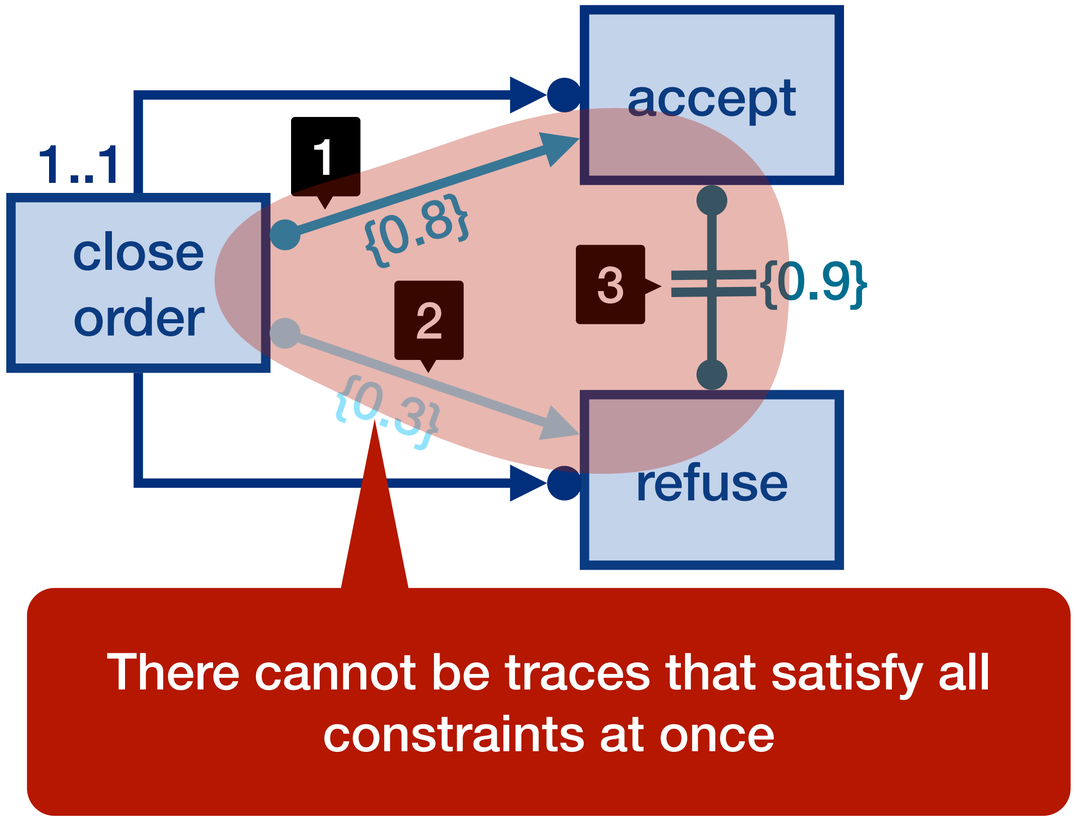
- All in all: up to 2^n scenarios, denoting different variants



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

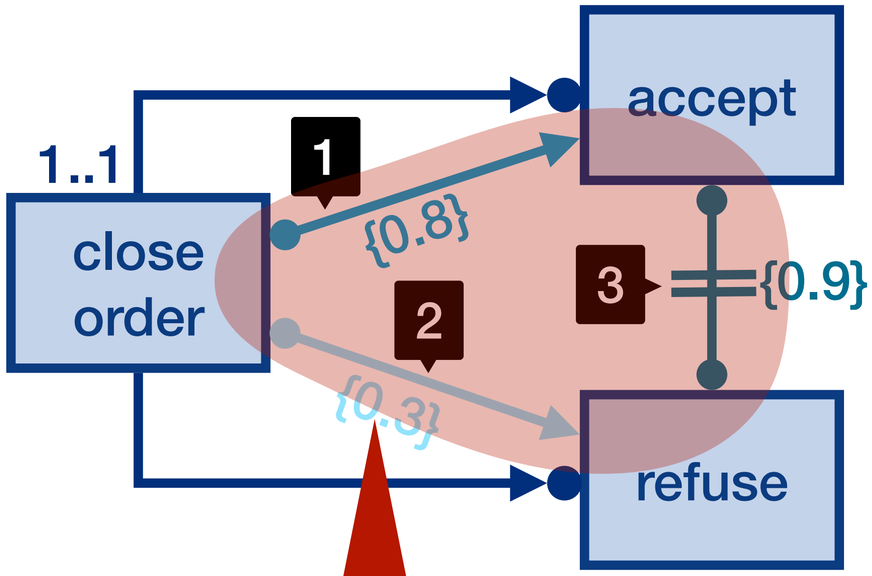
Interplay between logic and probabilities



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



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

There cannot be traces that satisfy all constraints at once

inconsistent
-> no satisfying trace
-> 0 probability!

Logical reasoning within scenarios

LTLf and automata to the rescue

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

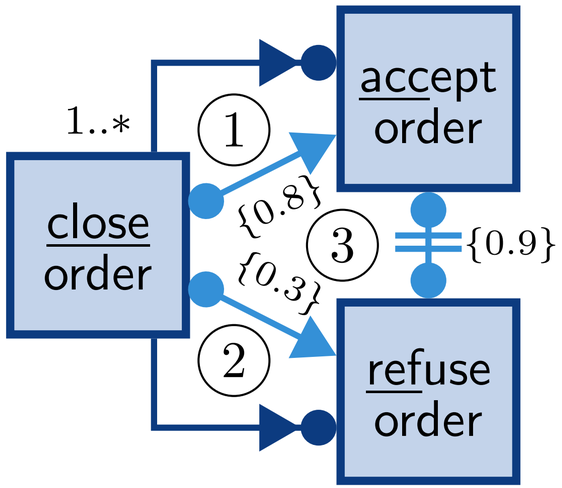
- Conjunction of formulae, one per constraint…
- Does the constraint hold in the scenario?
  - **Y** -> take its **LTLf process condition**
  - **N** -> take its **negation**

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

Reasoning via automata, as for standard LTLf

In our example…

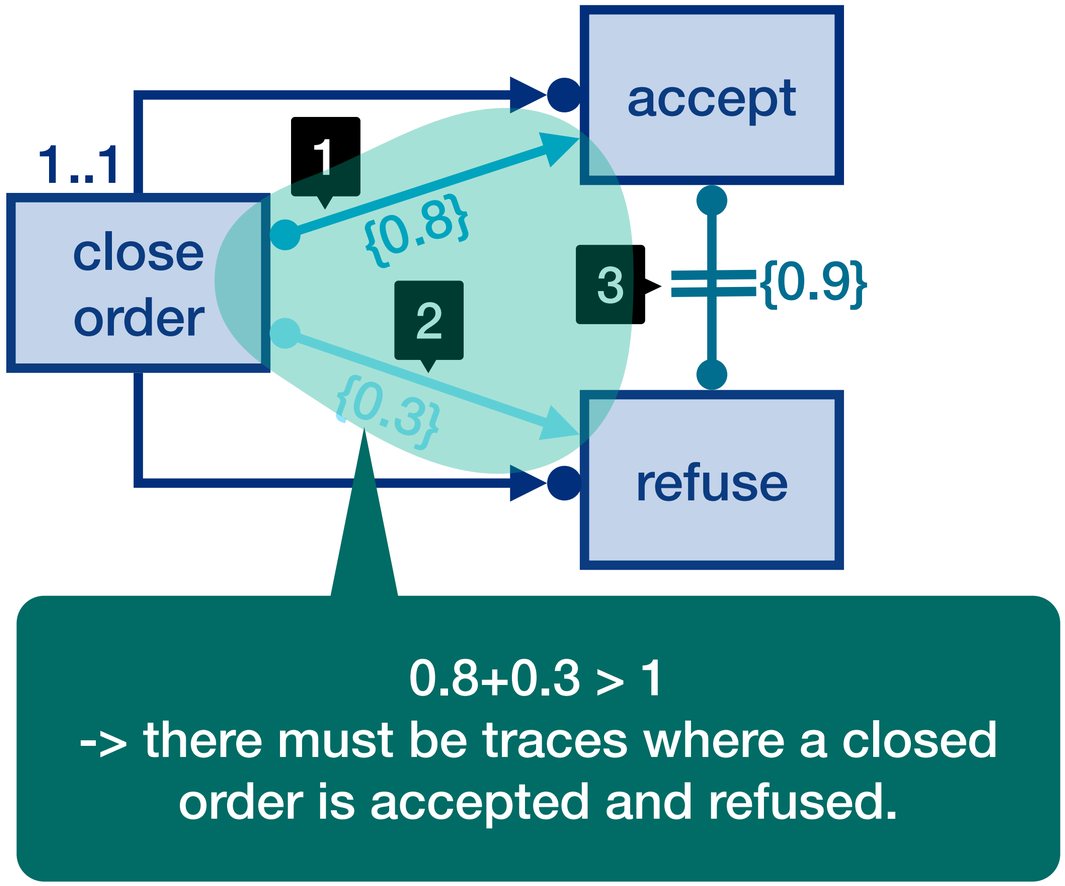
Which scenarios are consistent?



	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

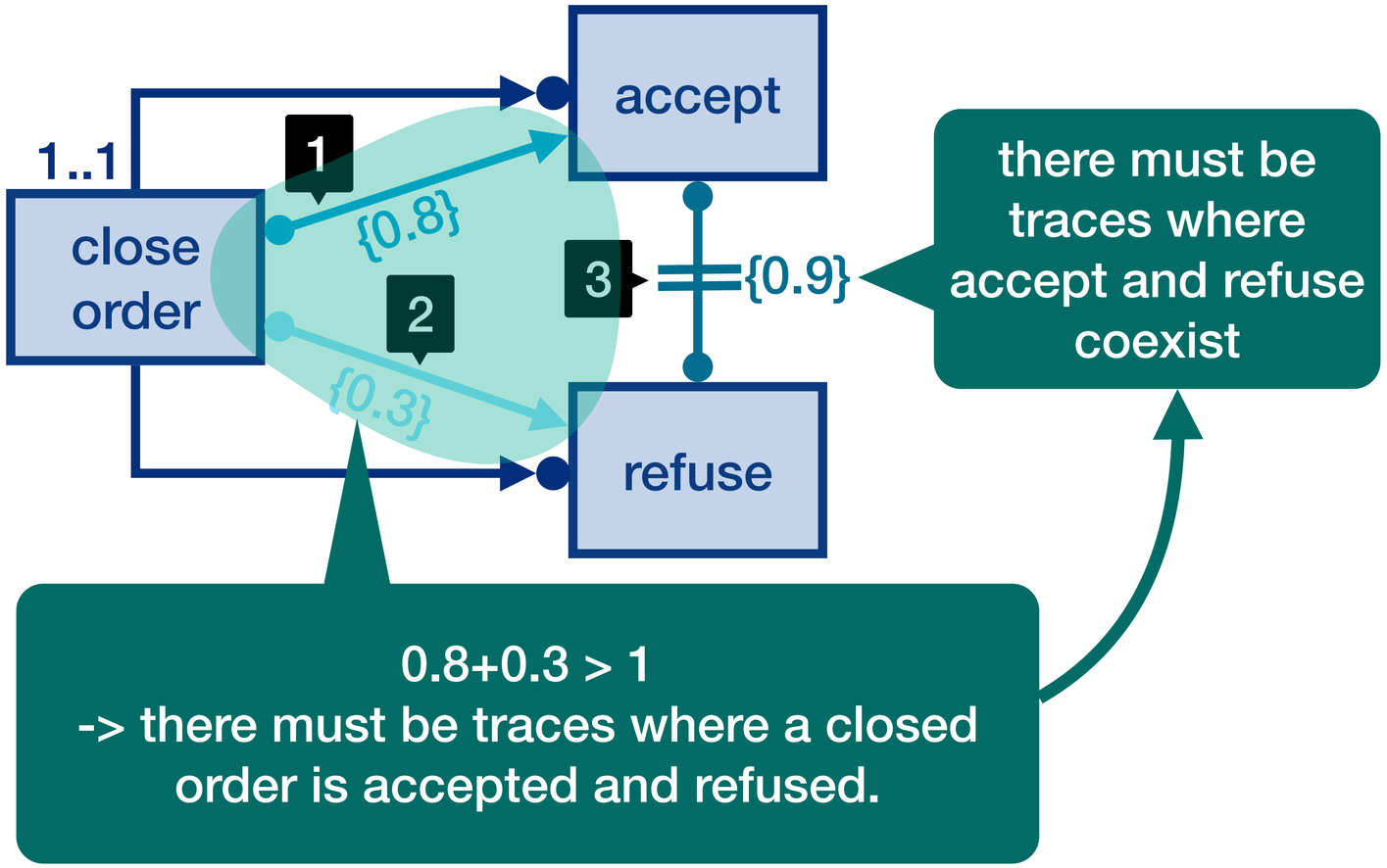
Interplay between logic and probabilities



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

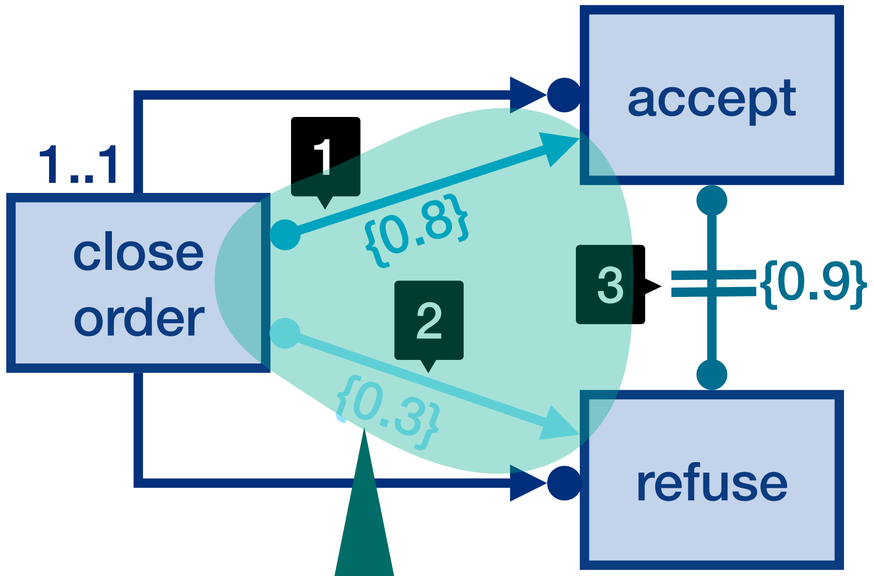
Interplay between logic and probabilities



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

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

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

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

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

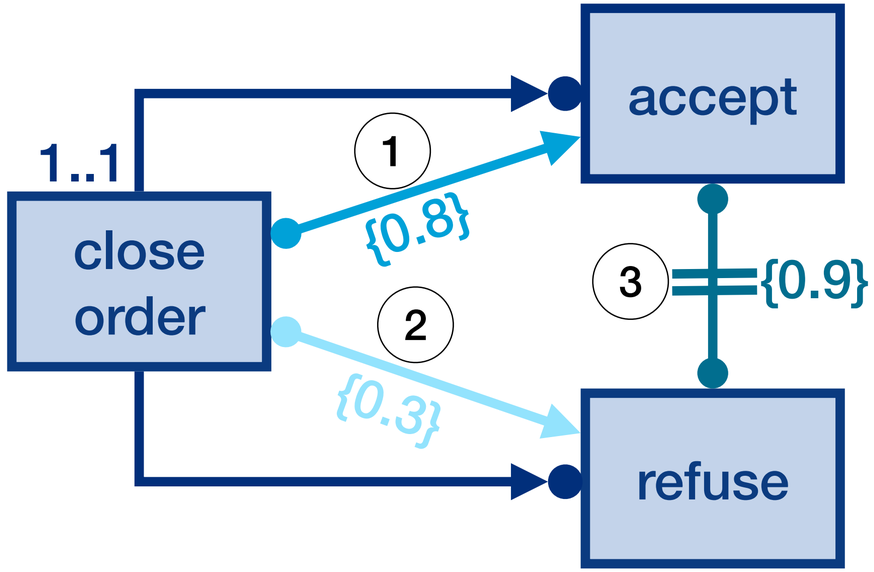
One solution
-> a fixed probability distribution

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

No solution
-> inconsistent specification

Computing probability distributions

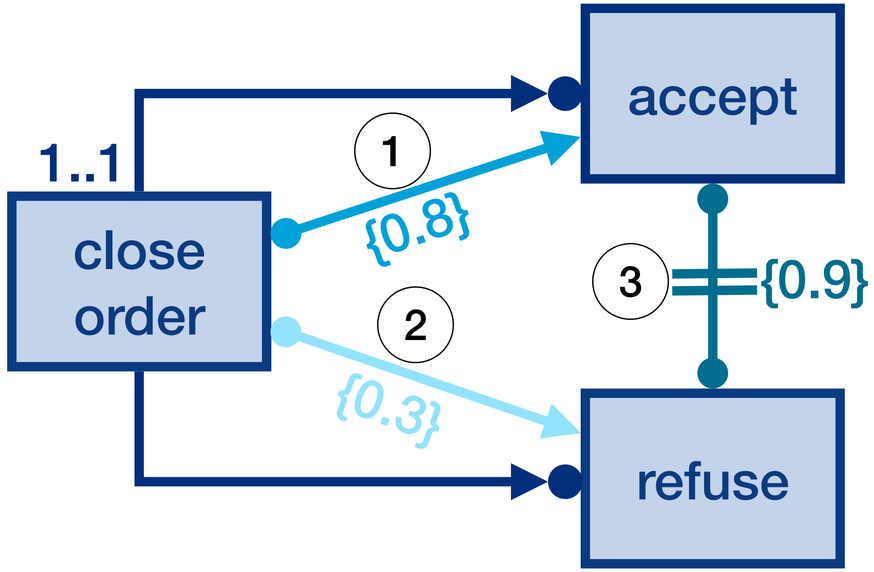
1. check for consistency



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

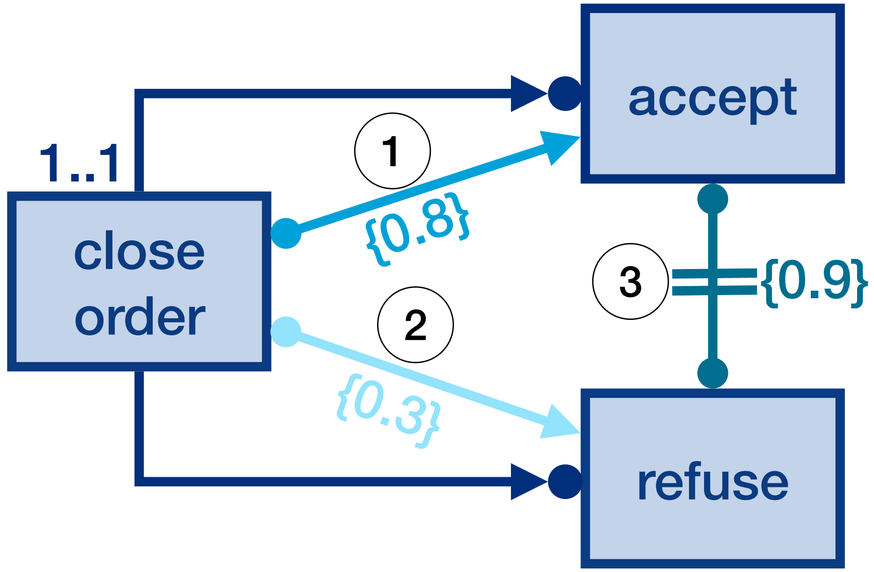
1. check for consistency



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

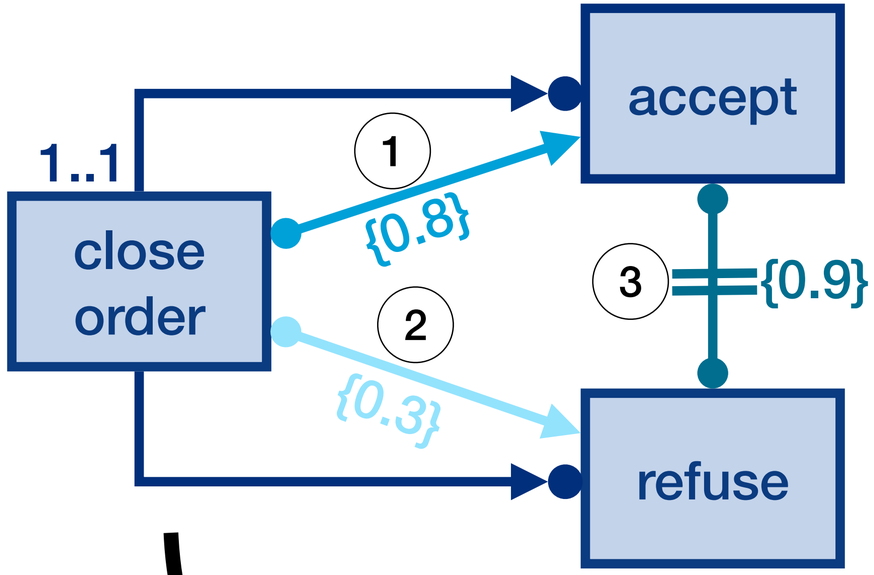
1. check for consistency



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

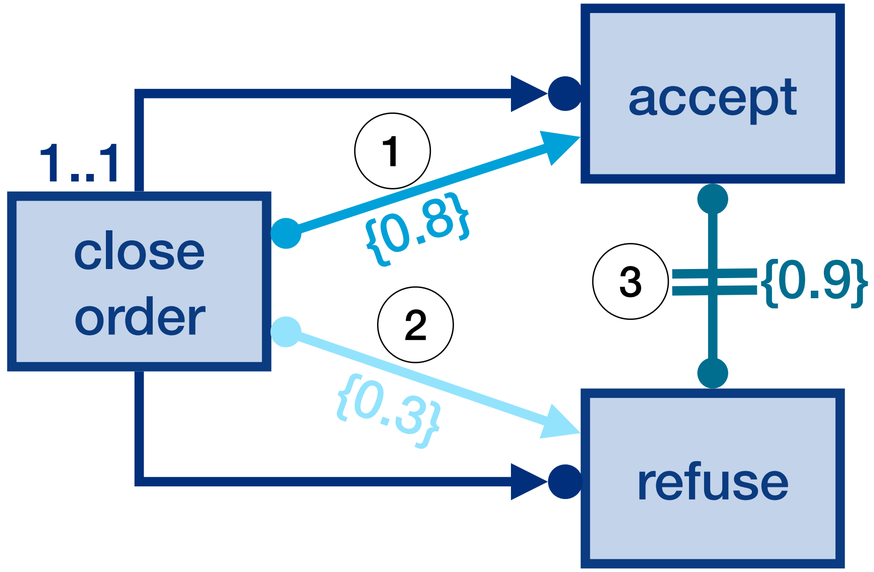


scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	
0	1	0	N	0
0	1	1	Y	
1	0	0	N	0
1	0	1	Y	
1	1	0	Y	
1	1	1	N	0

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

Computing probability distributions

3. solve

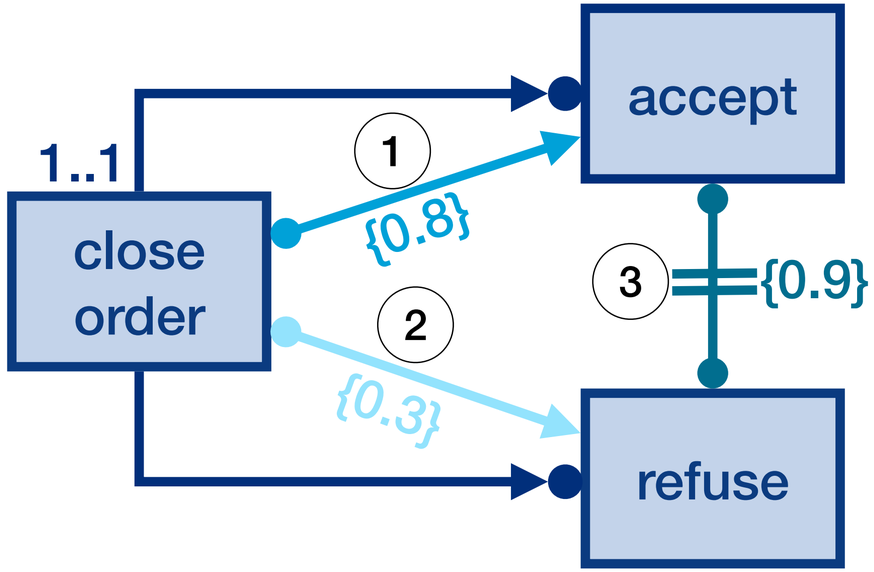


scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0

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

Computing probability distributions

3. solve

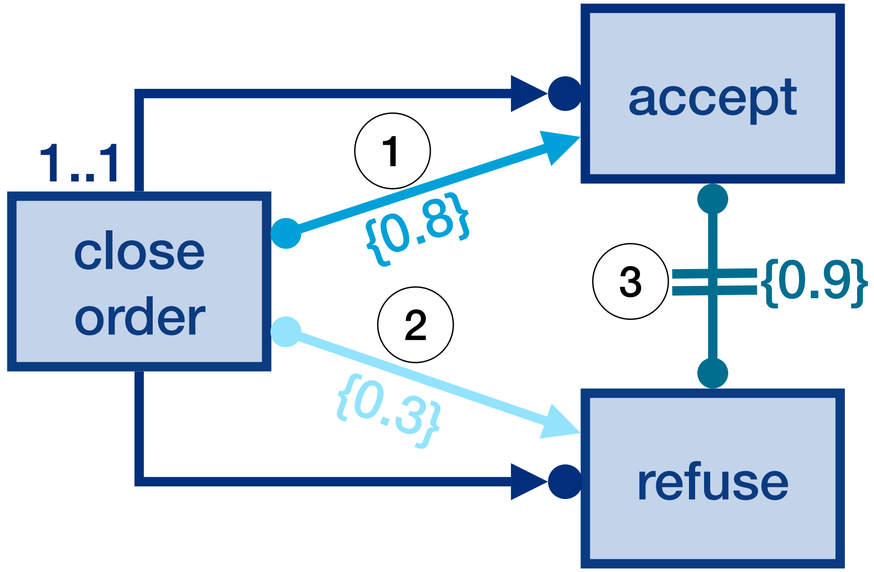


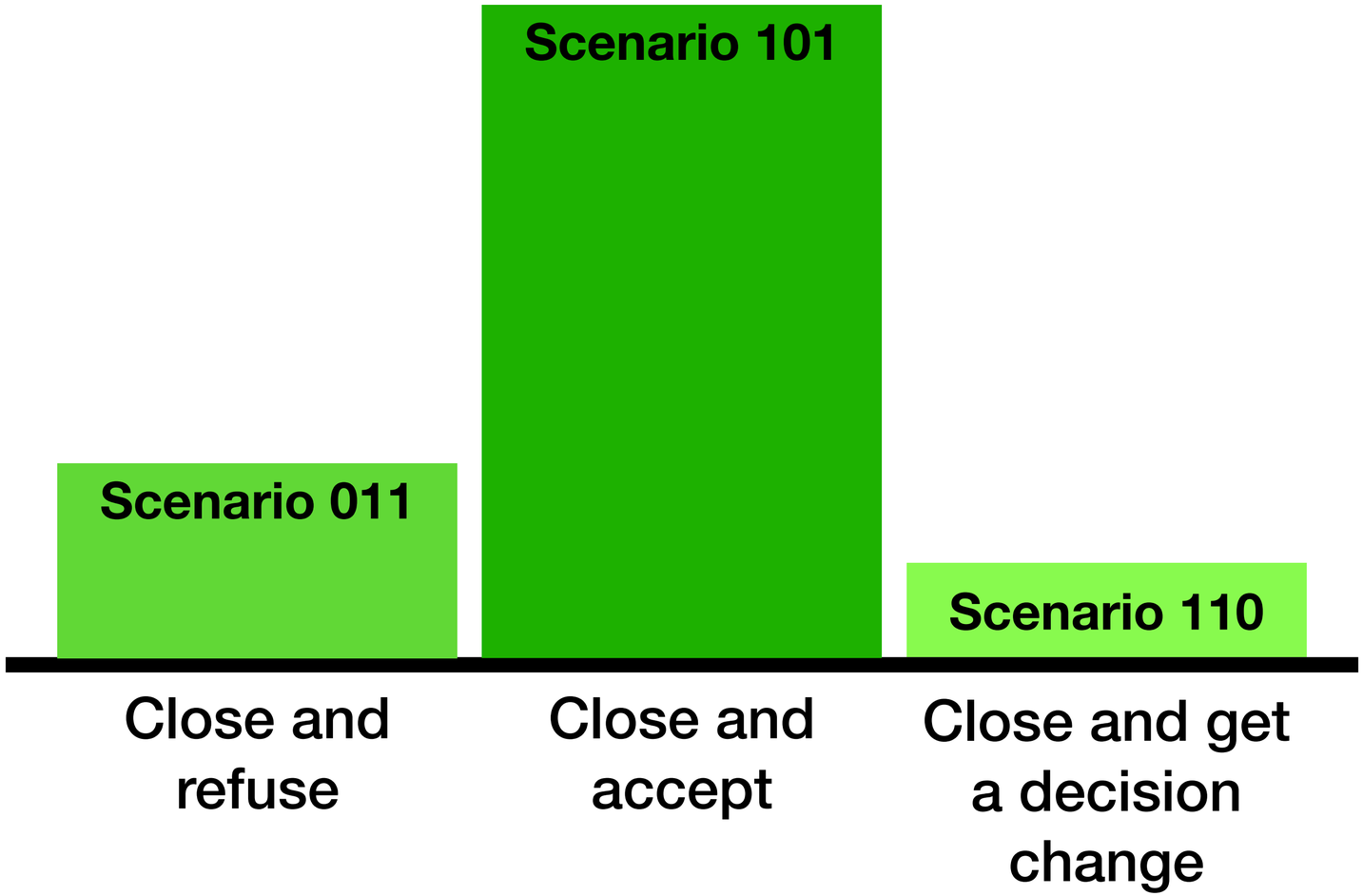
scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0

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

Computing probability distributions

3. solve

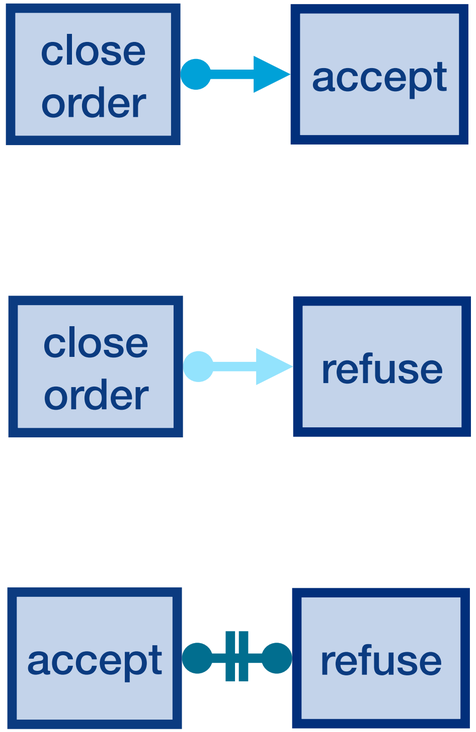




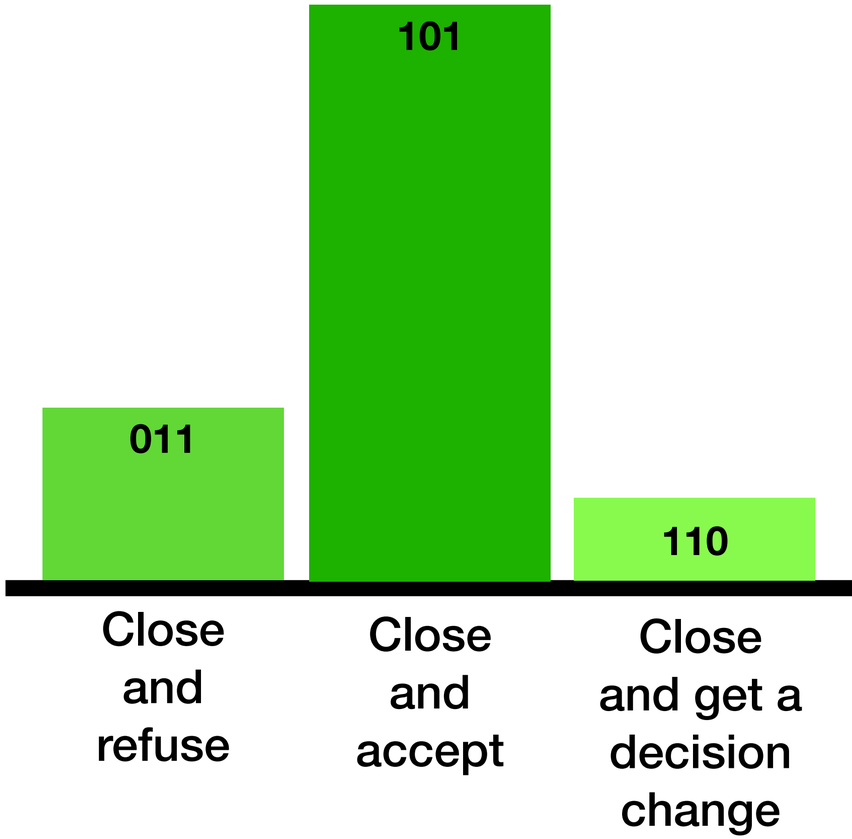
Scenarios in action

Conformance checking

<close order>

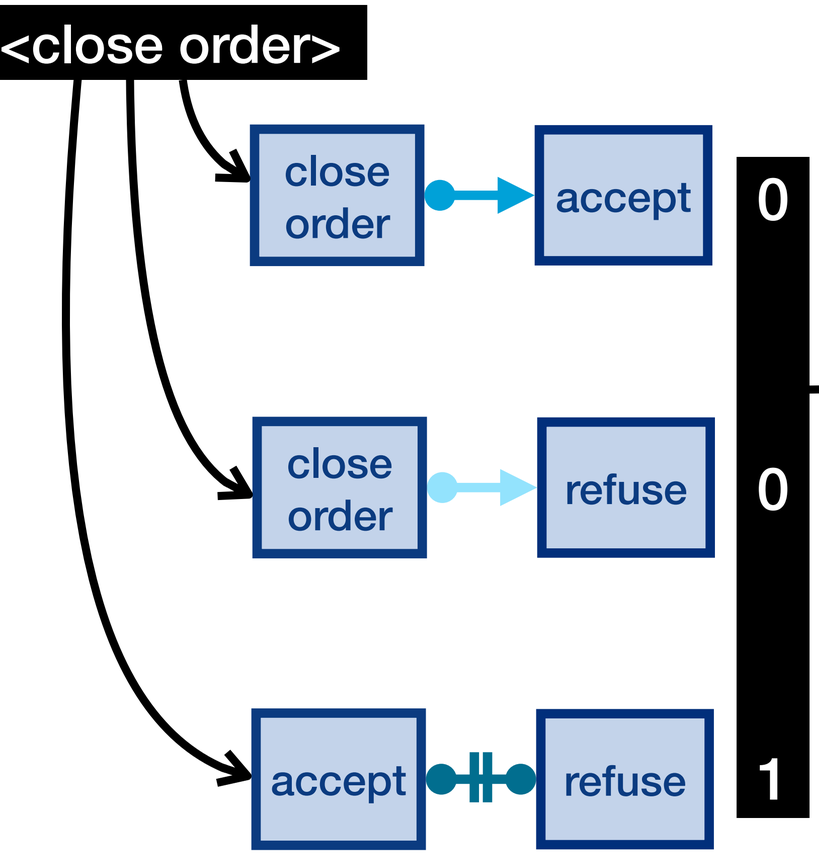


scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0



Scenarios in action

Conformance checking



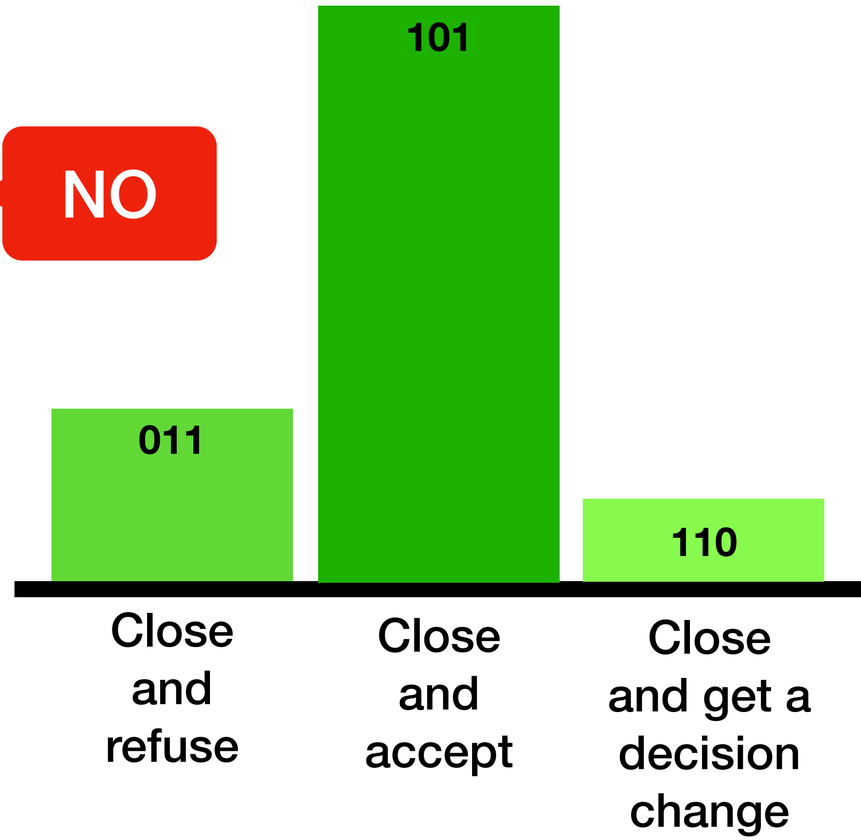
0

0

1

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

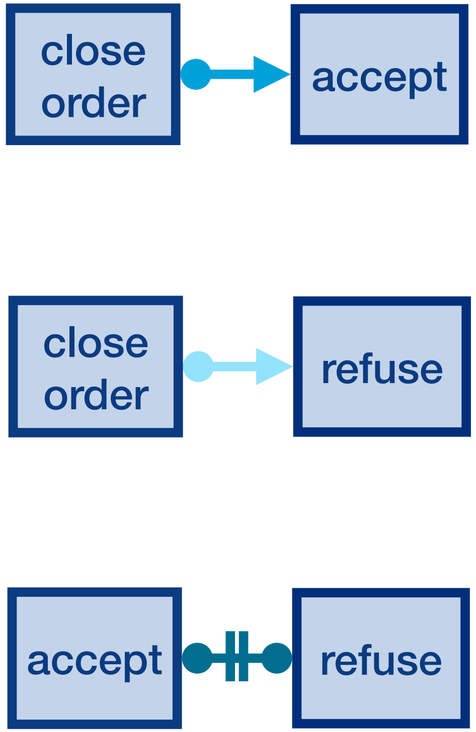
NO



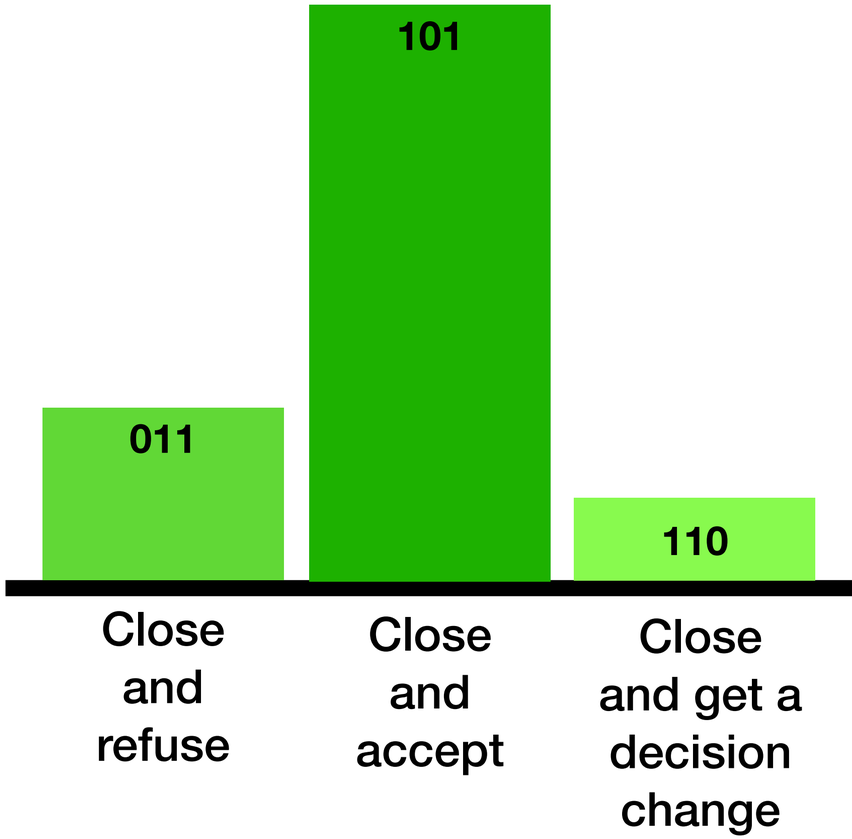
Scenarios in action

Conformance checking

<close order, accept, refuse>

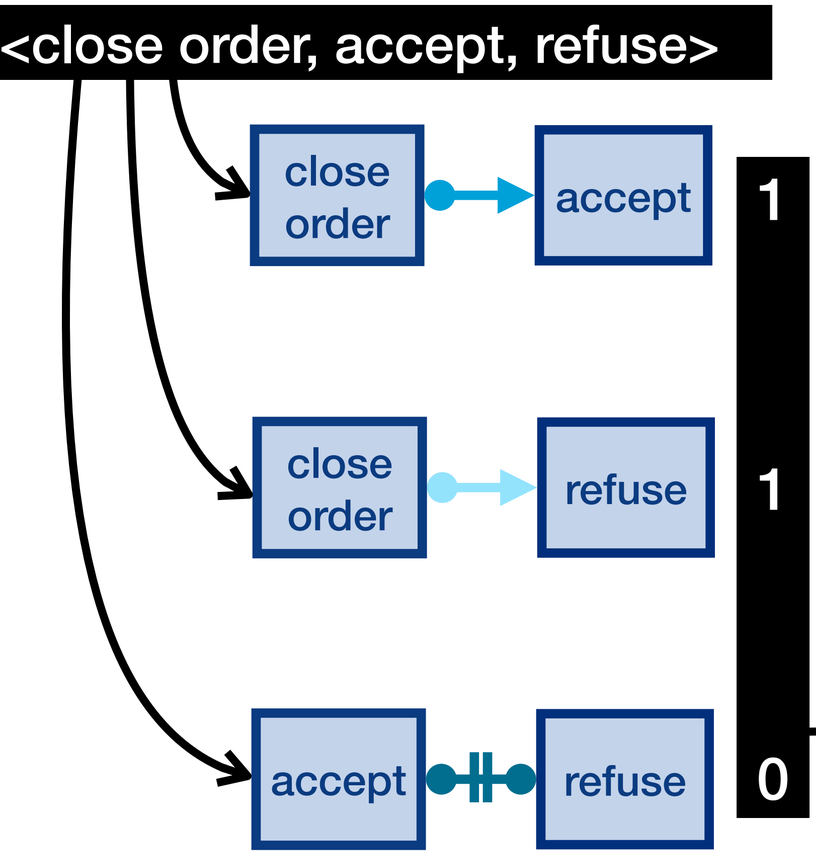


scenario			consistent?	probability
(1)	(2)	(3)		
0	0	0	N	0
0	0	1	Y	0
0	1	0	N	0
0	1	1	Y	0.2
1	0	0	N	0
1	0	1	Y	0.7
1	1	0	Y	0.1
1	1	1	N	0



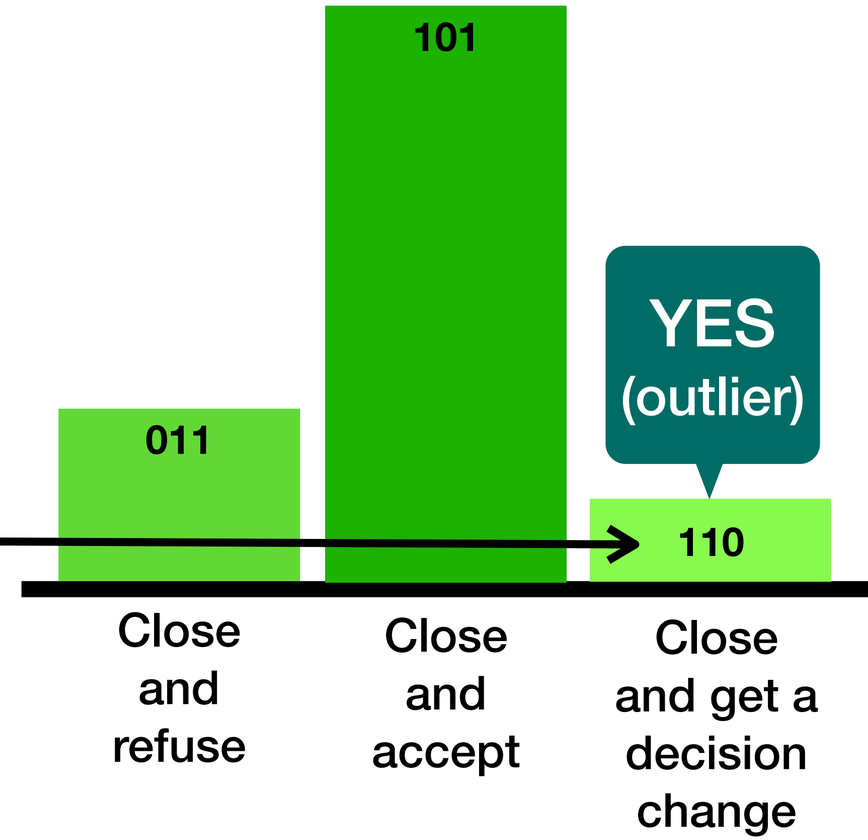
Scenarios in action

Conformance checking



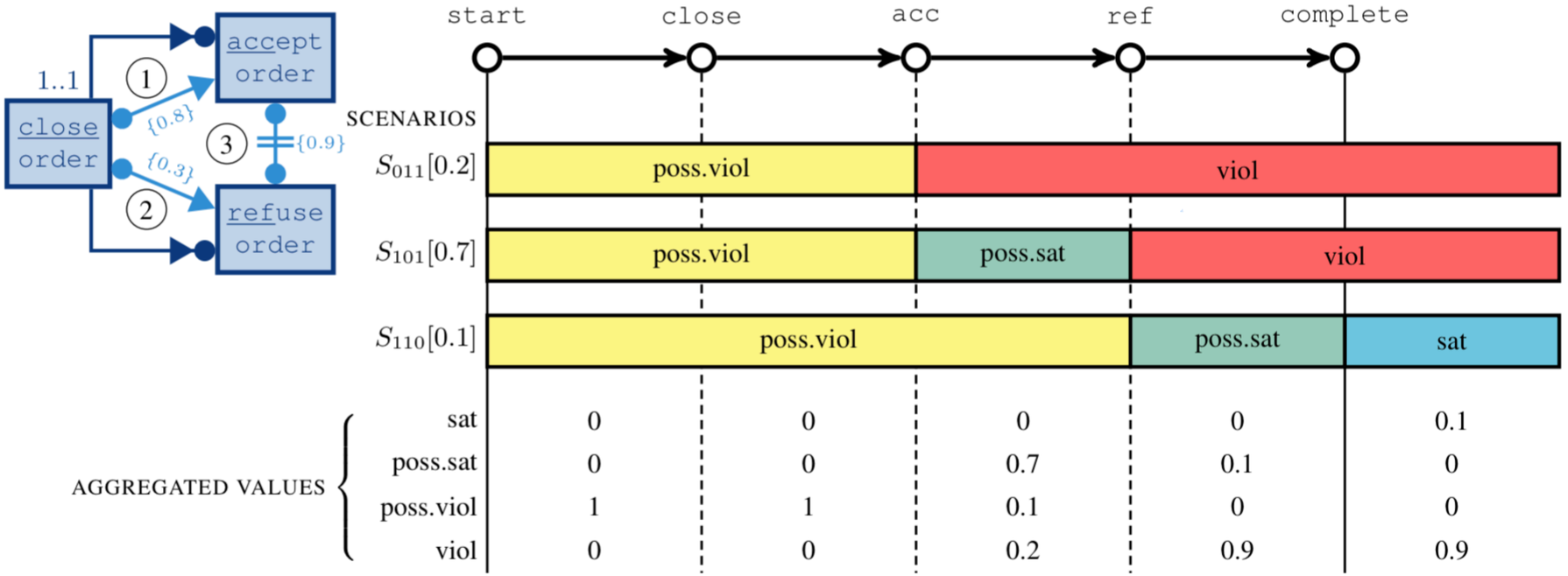
1
1
0

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

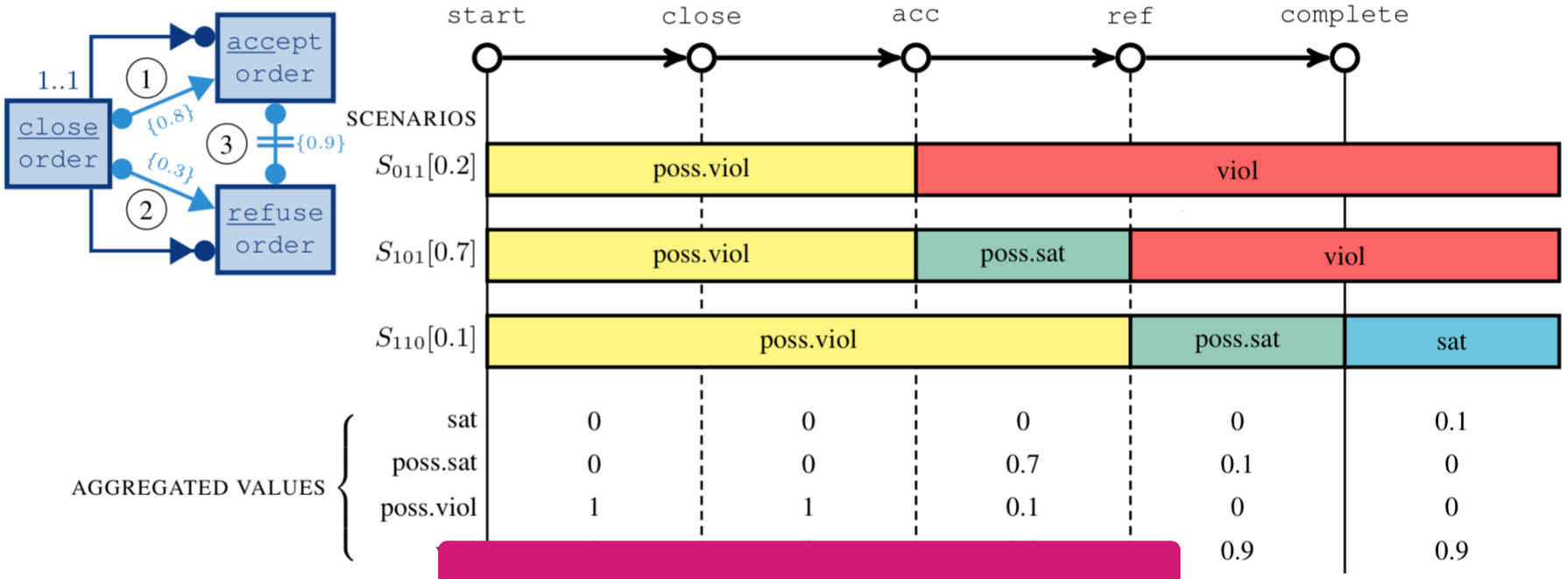
Probabilistic monitoring



- 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



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

Human interpretability is an interesting open challenge

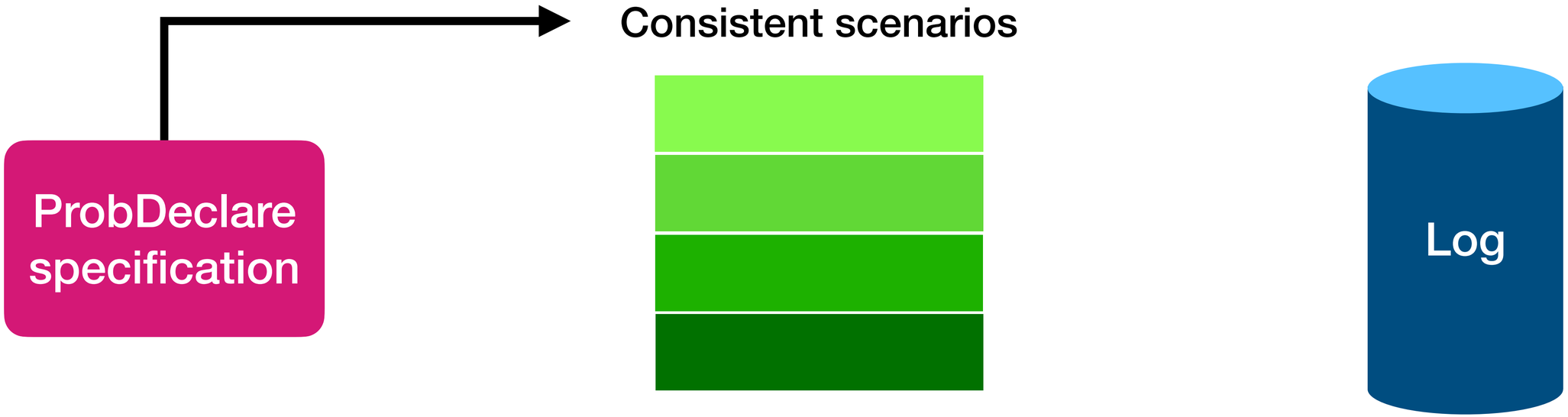
From traces to logs

Stochastic conformance (granularity: scenario)

ProbDeclare specification

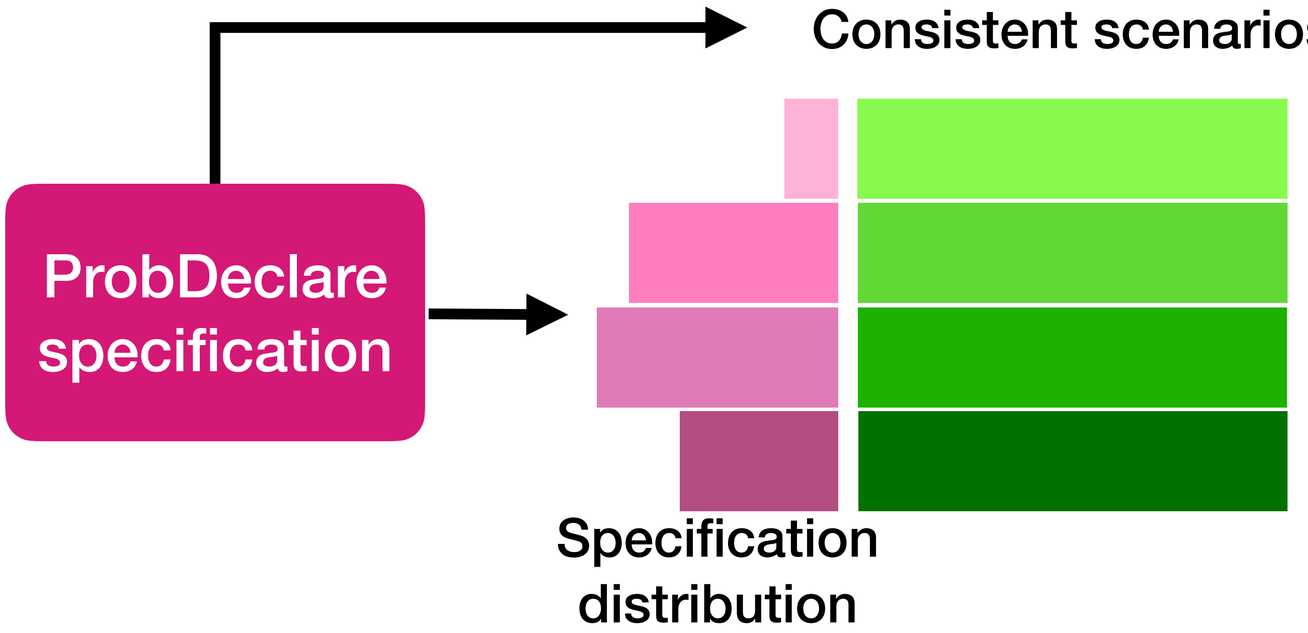
From traces to logs

Stochastic conformance (granularity: scenario)



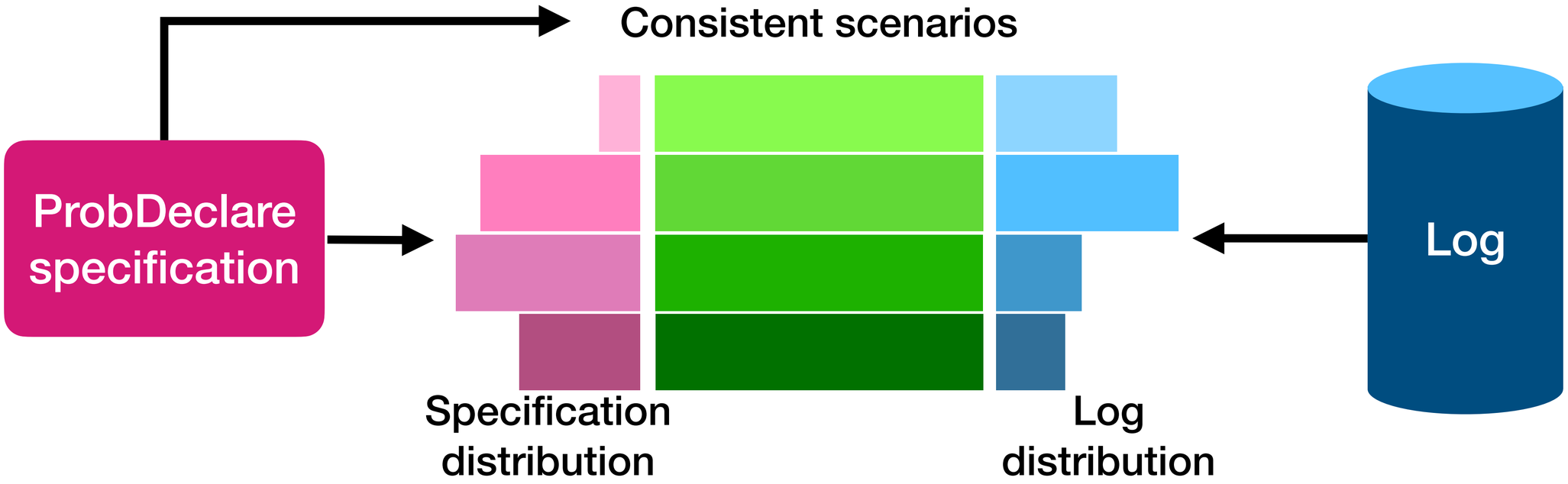
From traces to logs

Stochastic conformance (granularity: scenario)



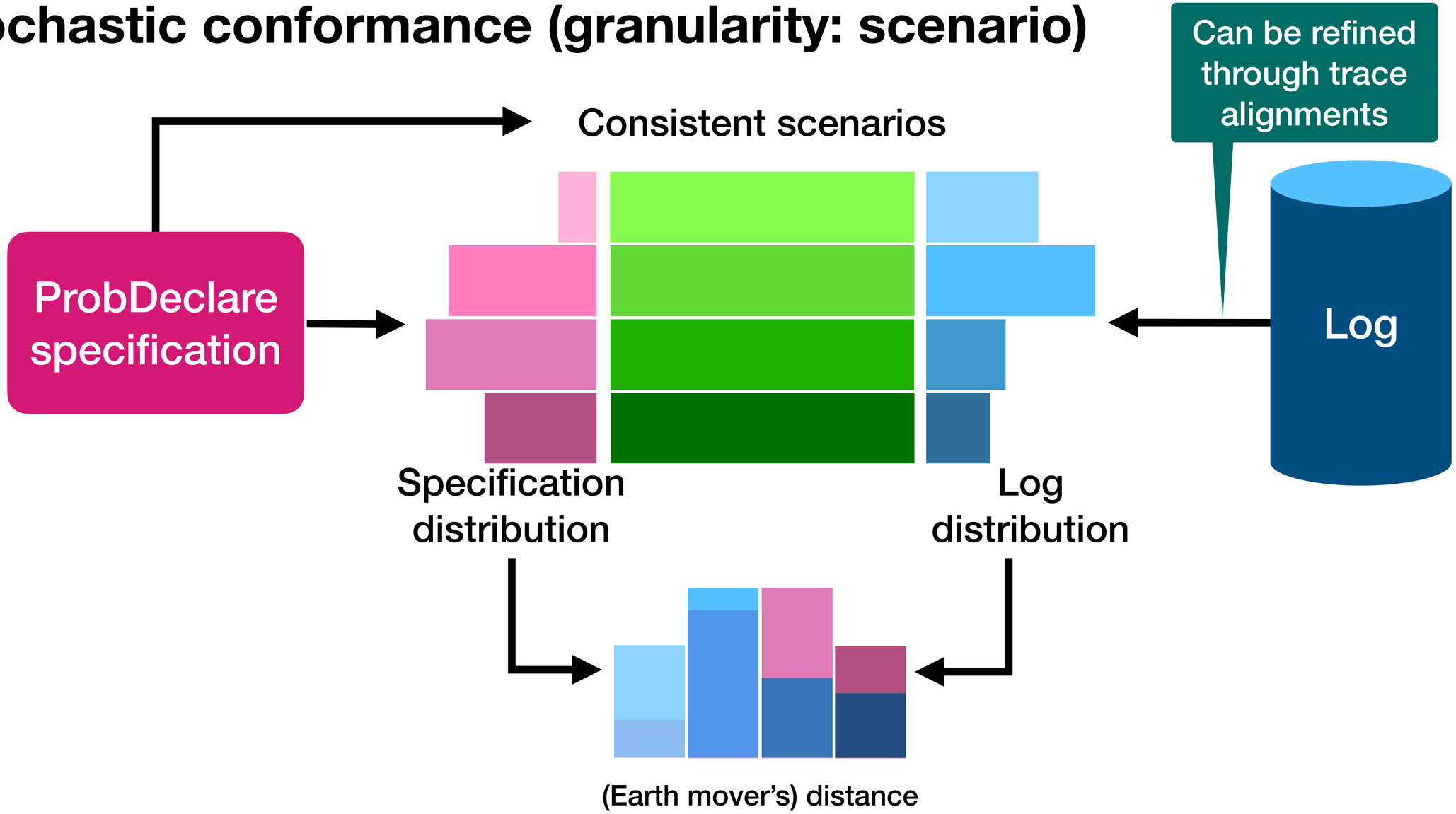
From traces to logs

Stochastic conformance (granularity: scenario)



From traces to logs

Stochastic conformance (granularity: scenario)



Declare discovery

All possible constraints grounded on the activities in the log

Declare discovery

Which to keep?

All possible constraints grounded on the activities in the log

Template algorithm for Declare discovery

[____,PMHandbook2022]

trace support: $\frac{\text{\# traces that "interestingly" satisfy the constraint}}{\text{total \# traces in the log}}$

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, …

Consistency guaranteed only for 100% trace-based support

Template algorithm for ProbDeclare discovery

[____,InfSys2022]

trace support: $\frac{\text{\# traces that "interestingly" satisfy the constraint}}{\text{total \# traces in the log}}$

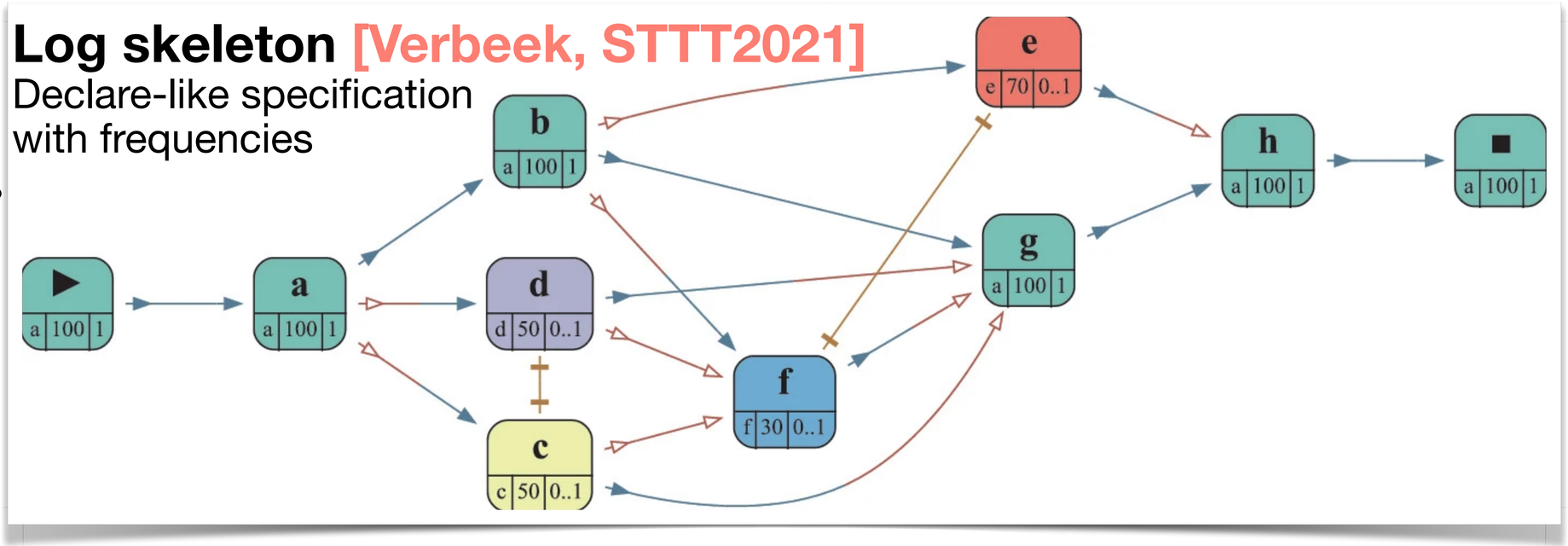
1. **Select templates** of interest
2. **Compute metrics** for corresponding constraints (grounded on log activities)
3. **Filter** based on minimum**/maximum** thresholds
4. **Redundant constraints**?
   - Keep **the most liberal** if **metrics are better** for it
   - Keep **the most restrictive** in case of **equal metrics**
5. ~~**Incompatible constraints**?~~
   - ~~Keep only the **one with better metrics**~~
5. **Use support as a basis for constraint probability**
   - **Consistency guaranteed by construction**
6. **Further processing** to ensure consistency, minimality, …

Idea: conversating on log skeletons

Discovery
[Verbeek, STTT2021]

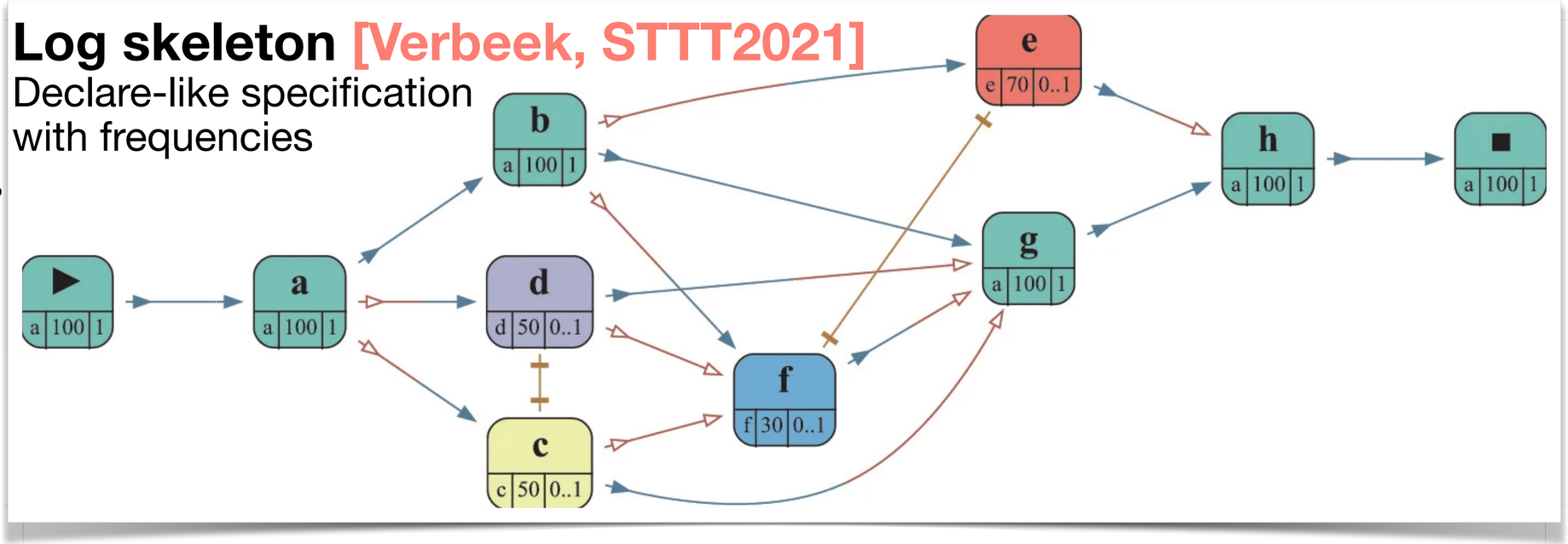
Log

Log skeleton [Verbeek, STTT2021]
Declare-like specification
with frequencies



Idea: conversating on log skeletons

Log skeleton [Verbeek, STTT2021]
Declare-like specification
with frequencies



Discovery
[Verbeek, STTT2021]

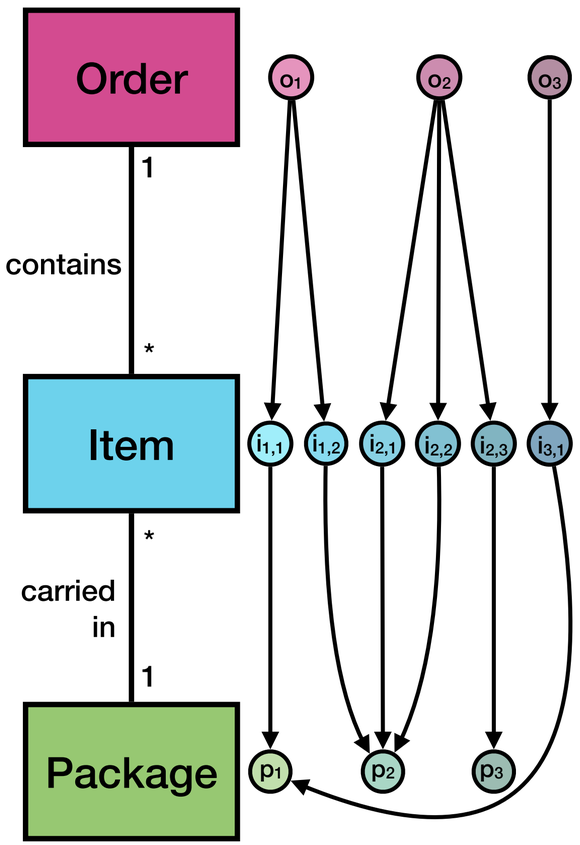
Log

Reasoning on constraints and their frequencies

Processes are not flat

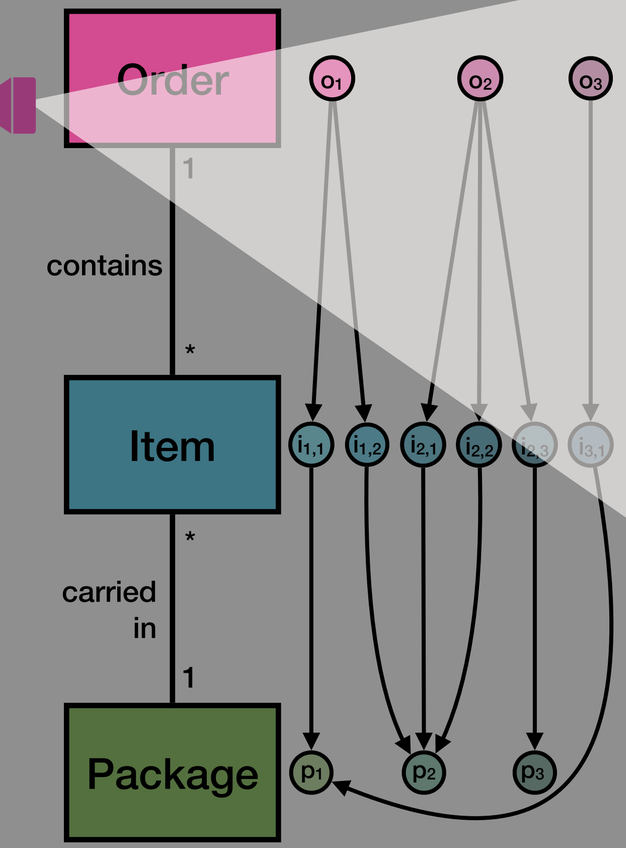
timestamp	overall log
2019-09-22 10:00:00	create order o_1
2019-09-22 10:01:00	add item $i_{1,1}$ to order o_1
2019-09-23 09:20:00	create order o_2
2019-09-23 09:34:00	add item $i_{2,1}$ to order o_2
2019-09-23 11:33:00	create order o_3
2019-09-23 11:40:00	add item $i_{3,1}$ to order o_3
2019-09-23 12:27:00	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

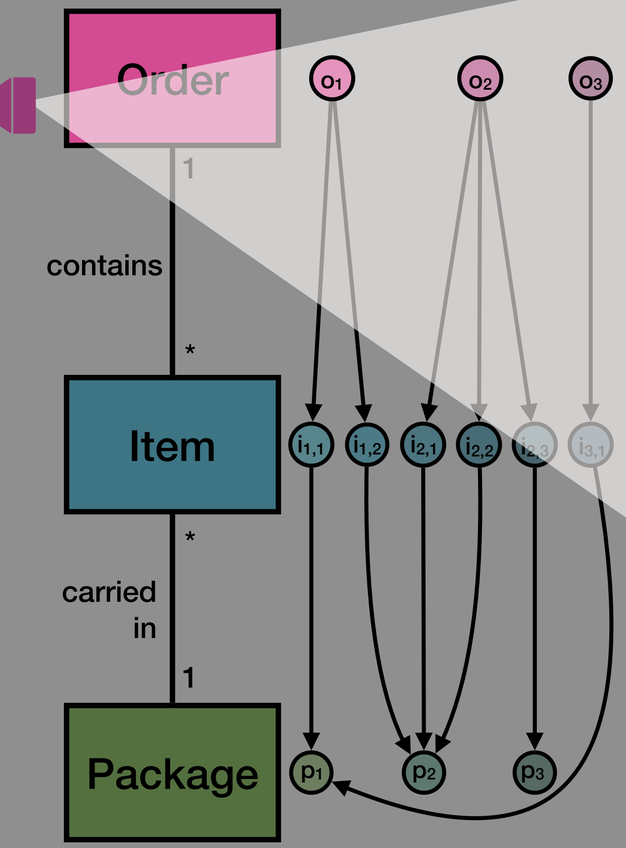


timestamp	overall log
2019-09-22 10:00:00	create order o_1
2019-09-22 10:01:00	add item $i_{1,1}$ to order o_1
2019-09-23 09:20:00	create order o_2
2019-09-23 09:34:00	add item $i_{2,1}$ to order o_2
2019-09-23 11:33:00	create order o_3
2019-09-23 11:40:00	add item $i_{3,1}$ to order o_3
2019-09-23 12:27:00	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	

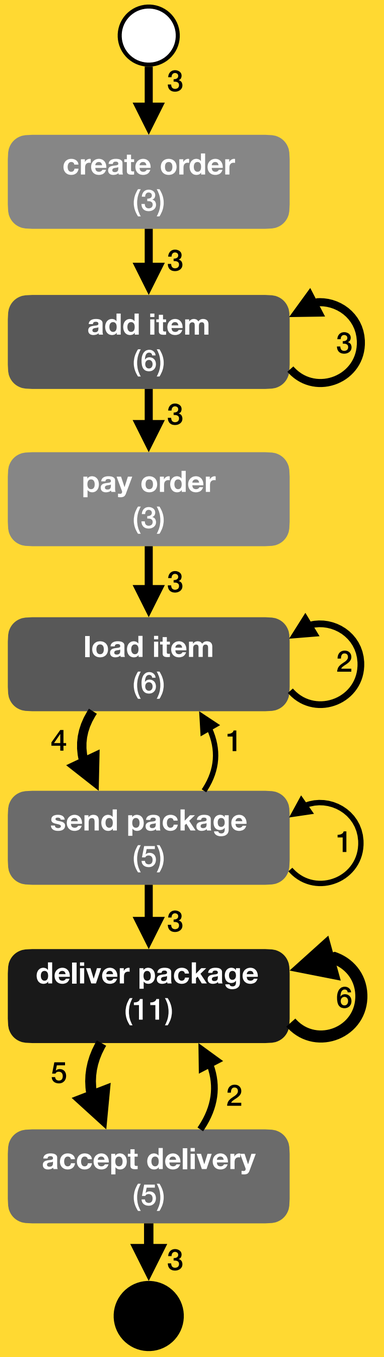
Processes are not flat



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

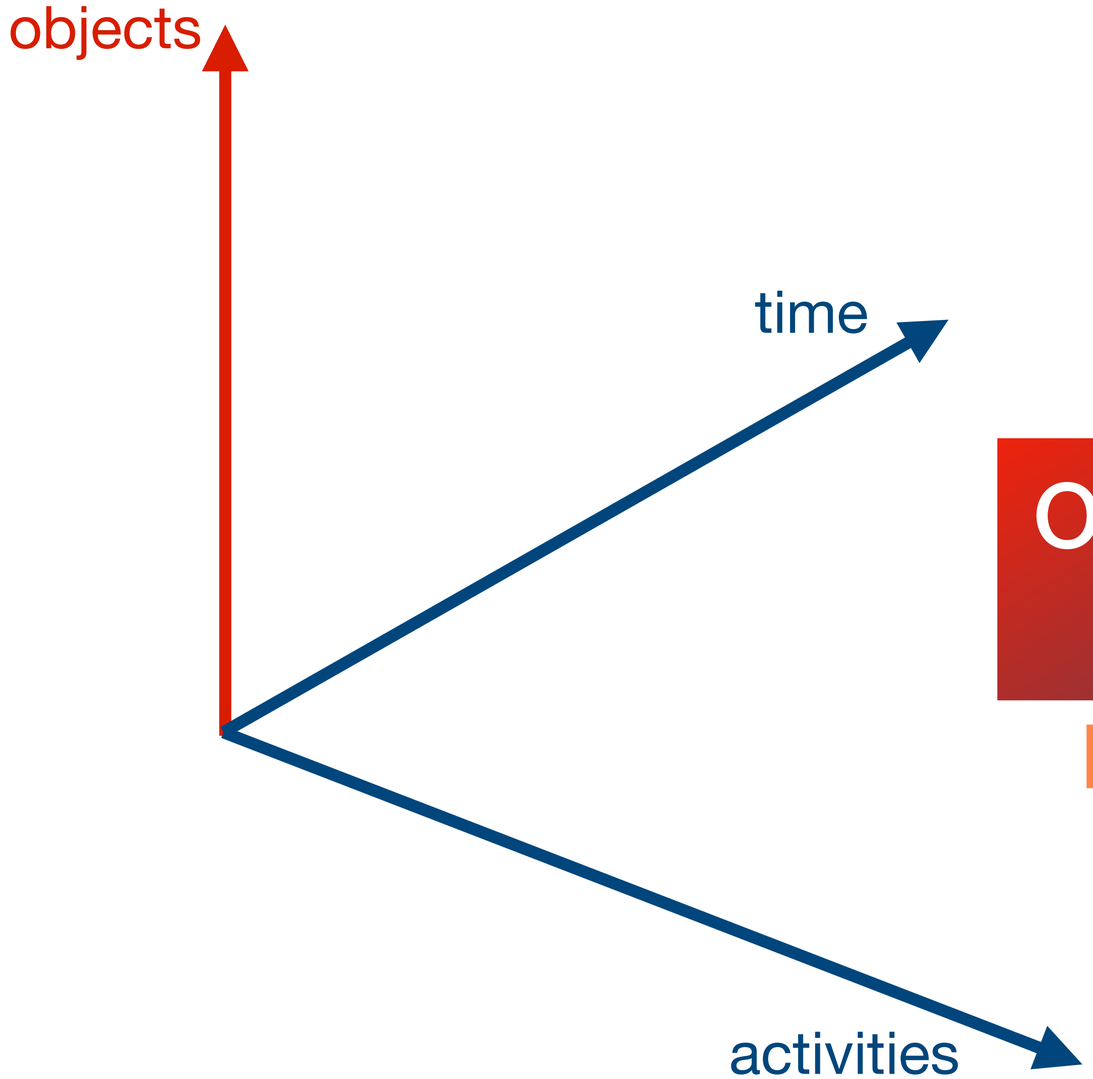
event log for orders

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	



Dealing with multiple objects

Need of a 3D model



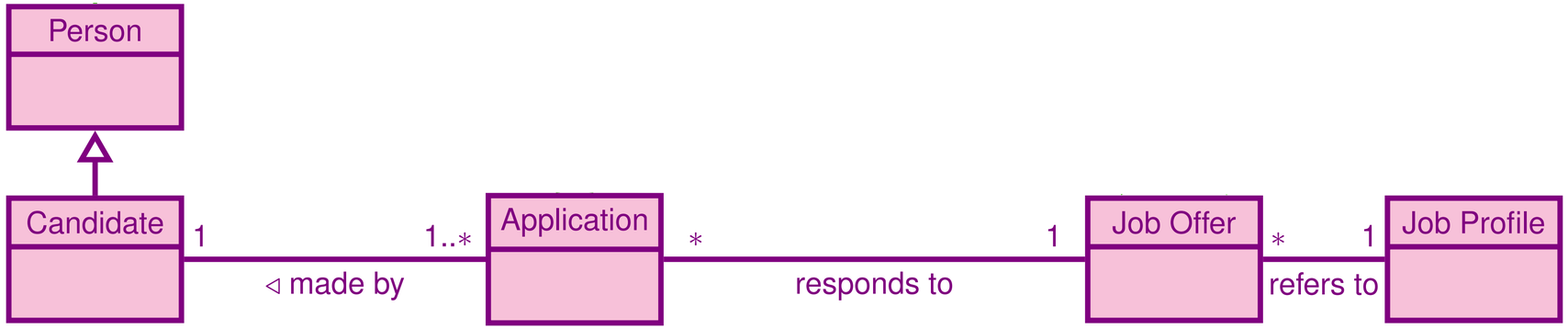
Object-centric behavioral constraints

[____,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)

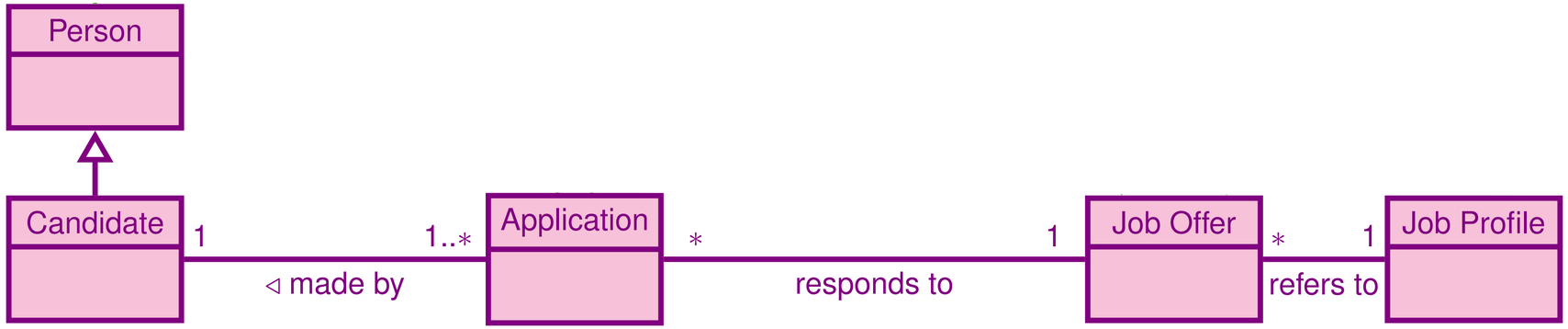


Object-centric behavioral constraints

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**.

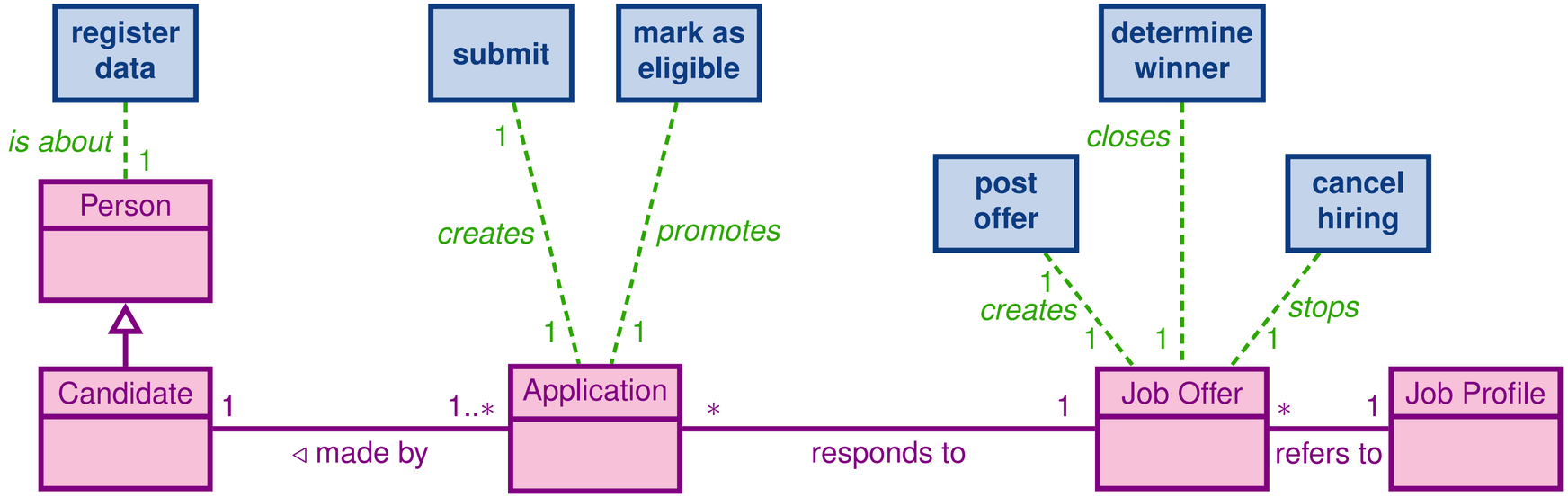


Object-centric behavioral constraints

Dimension 2: activities

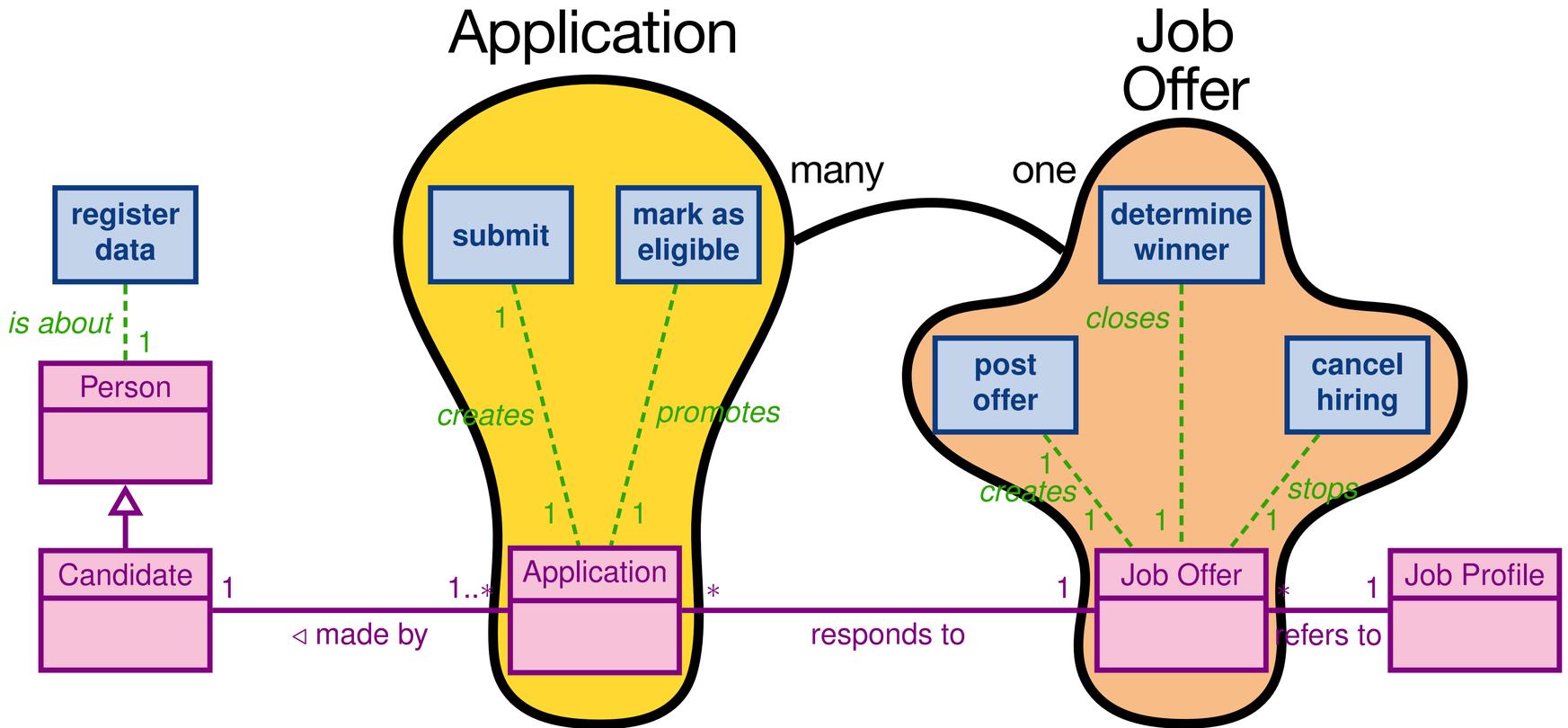
- **activities**
- **activity-class relationship types**
- **multiplicities**

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



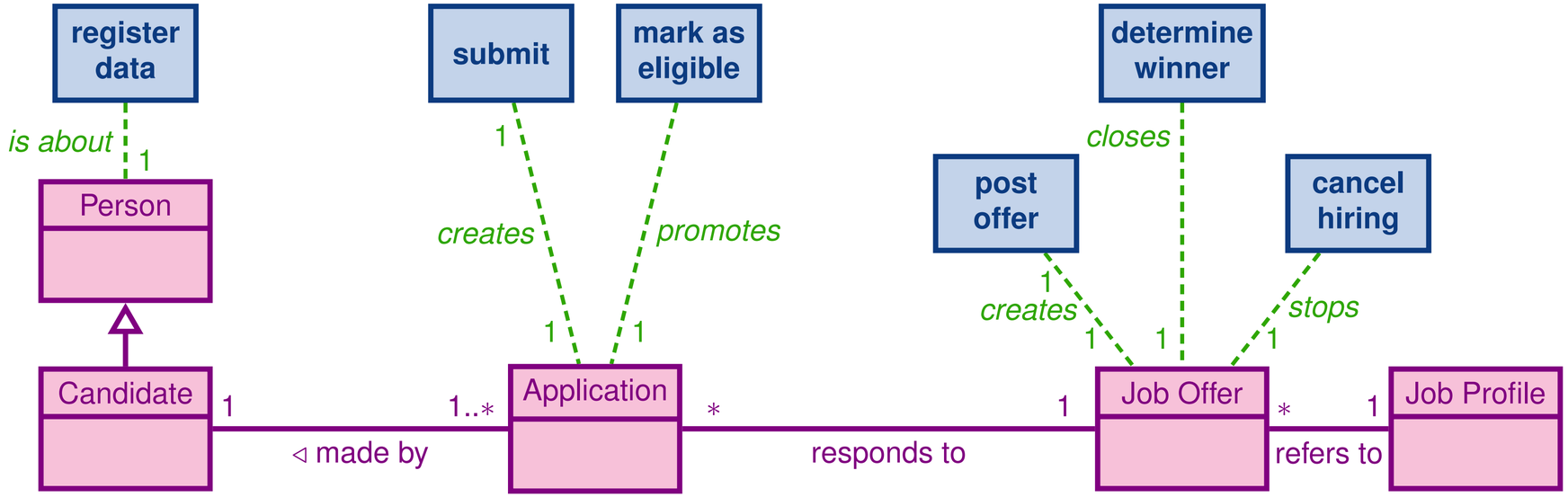
Object-centric behavioral constraints

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

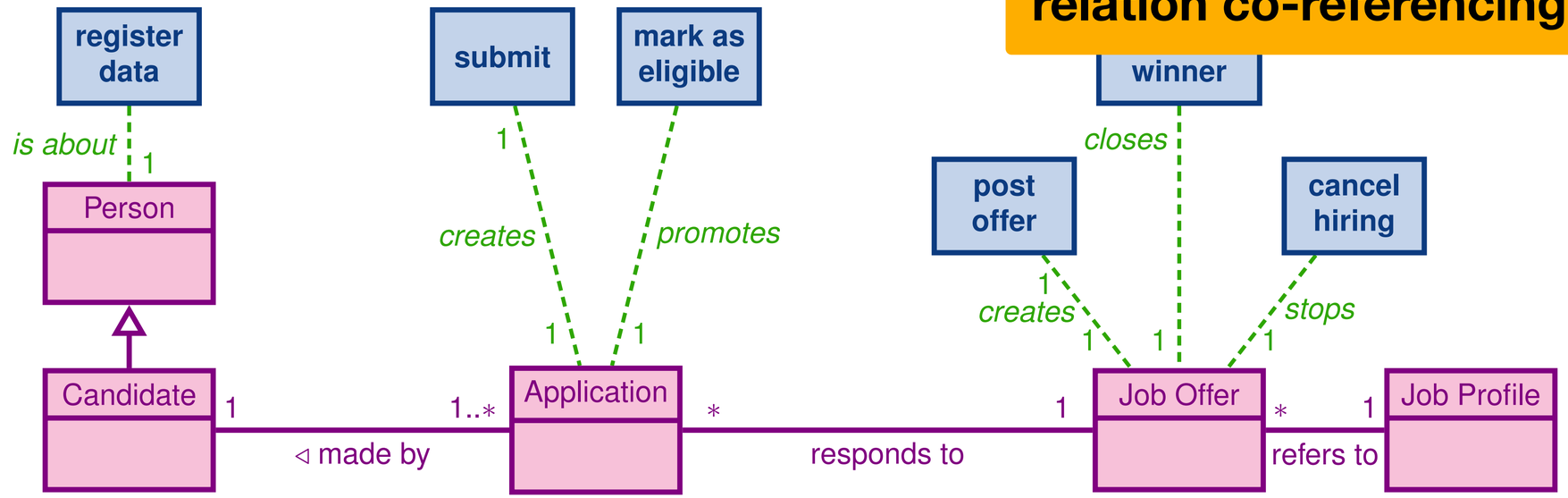
- **constraints…**
- **…with data co-referencing**

object co-referencing

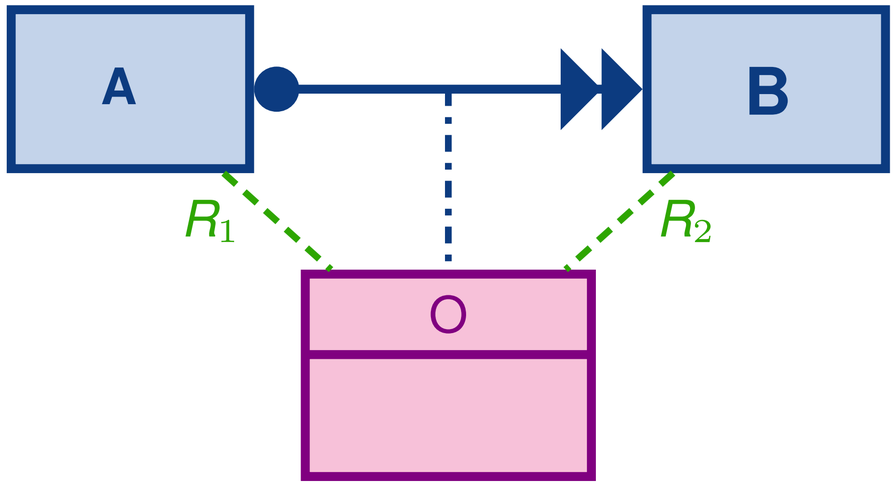
A Job Offer *closed by* a **determine winner** task *cannot* be *stopped* by executing the **cancel hiring** task (and vice-versa).

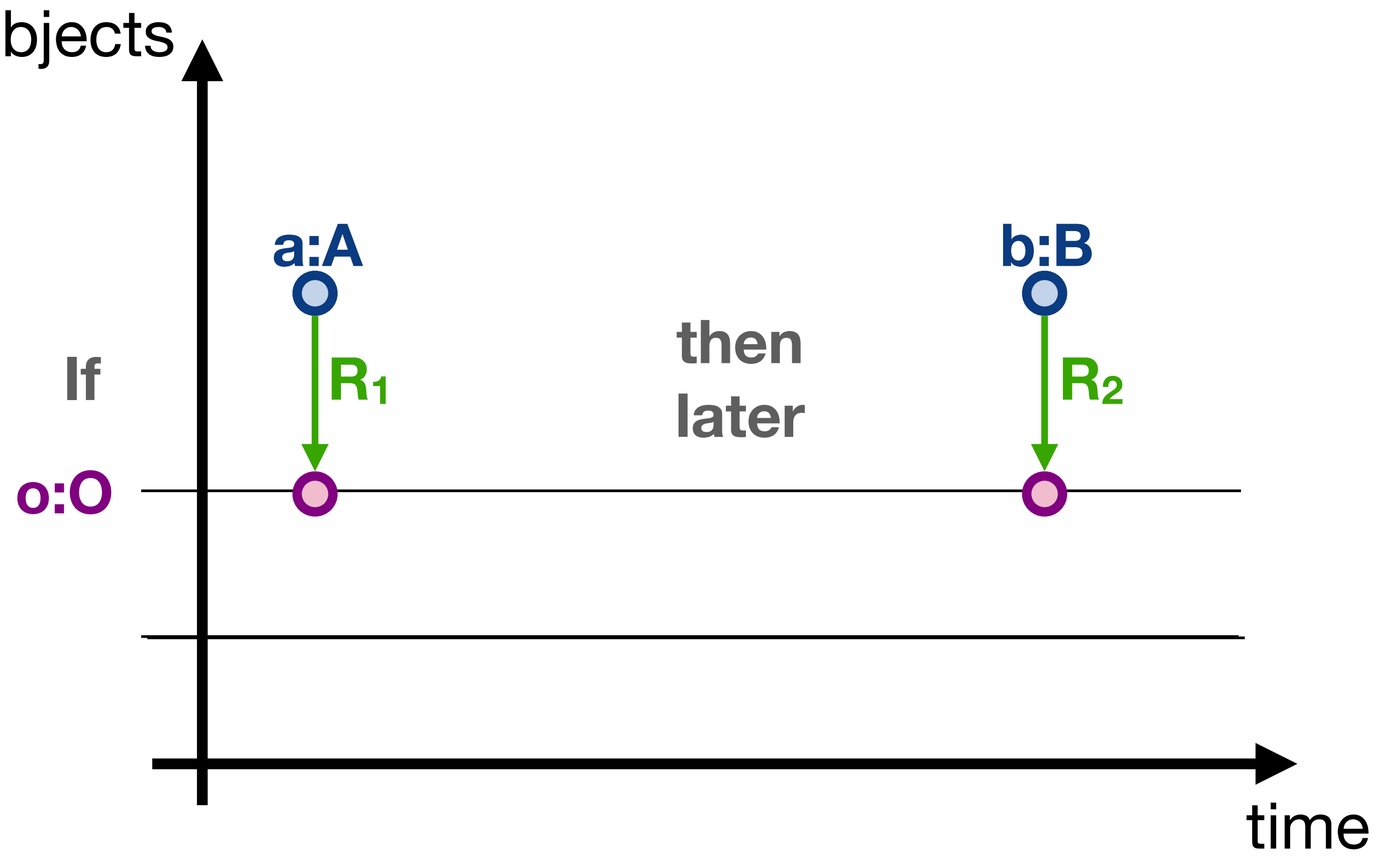
An Application can be **submitted** only if, *beforehand*, the **data** *about* the Candidate who made *that* Application have been **registered**.

relation co-referencing

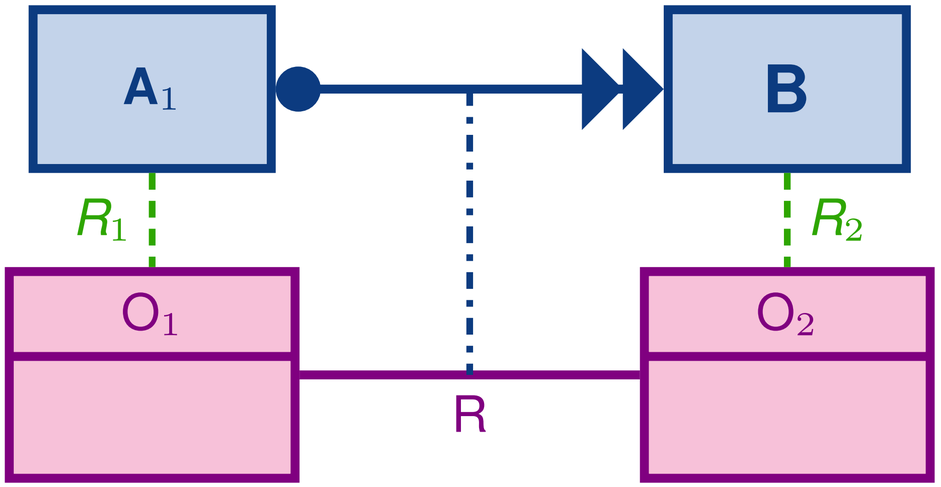


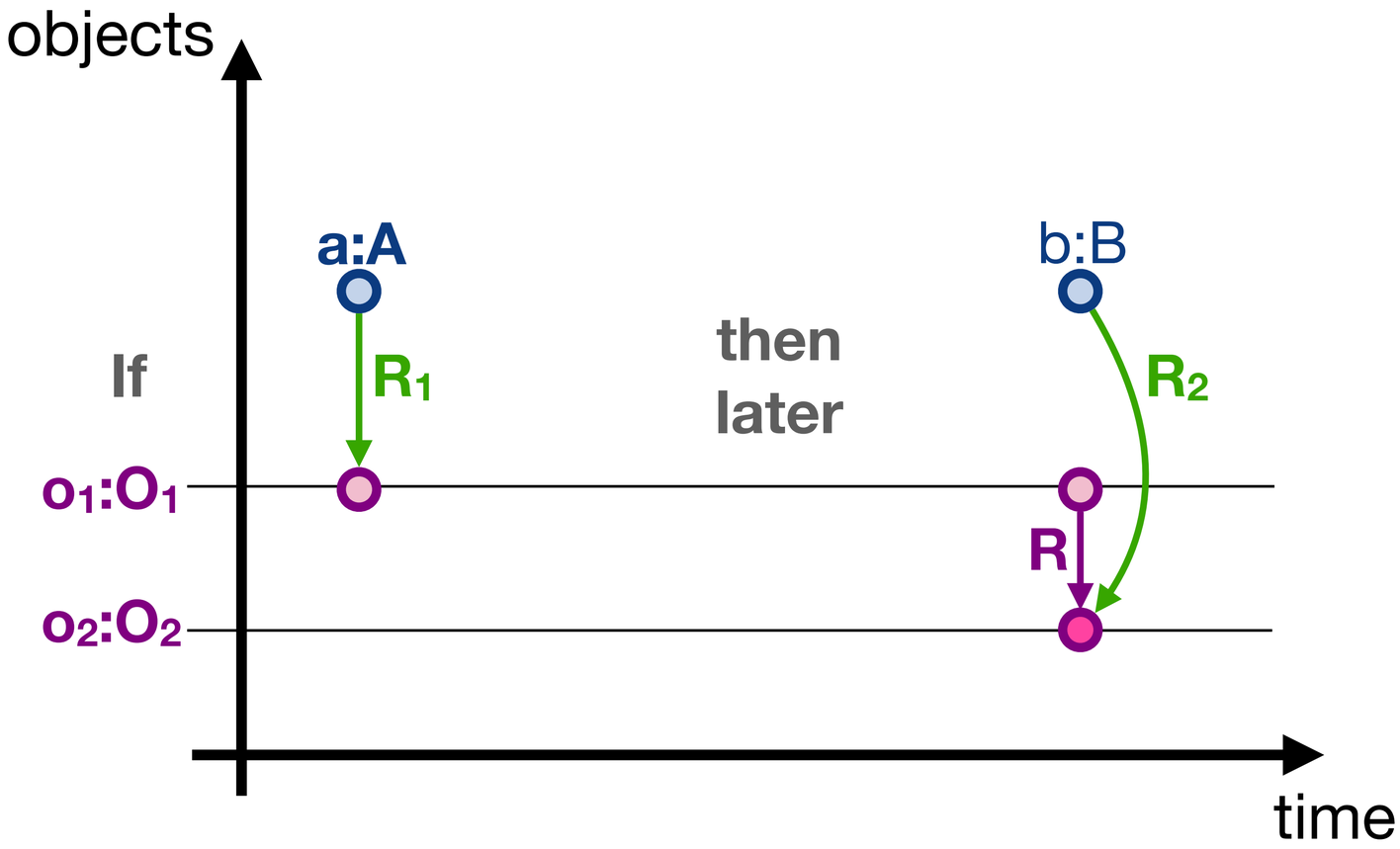
Object co-referencing on response





Relation co-referencing on response





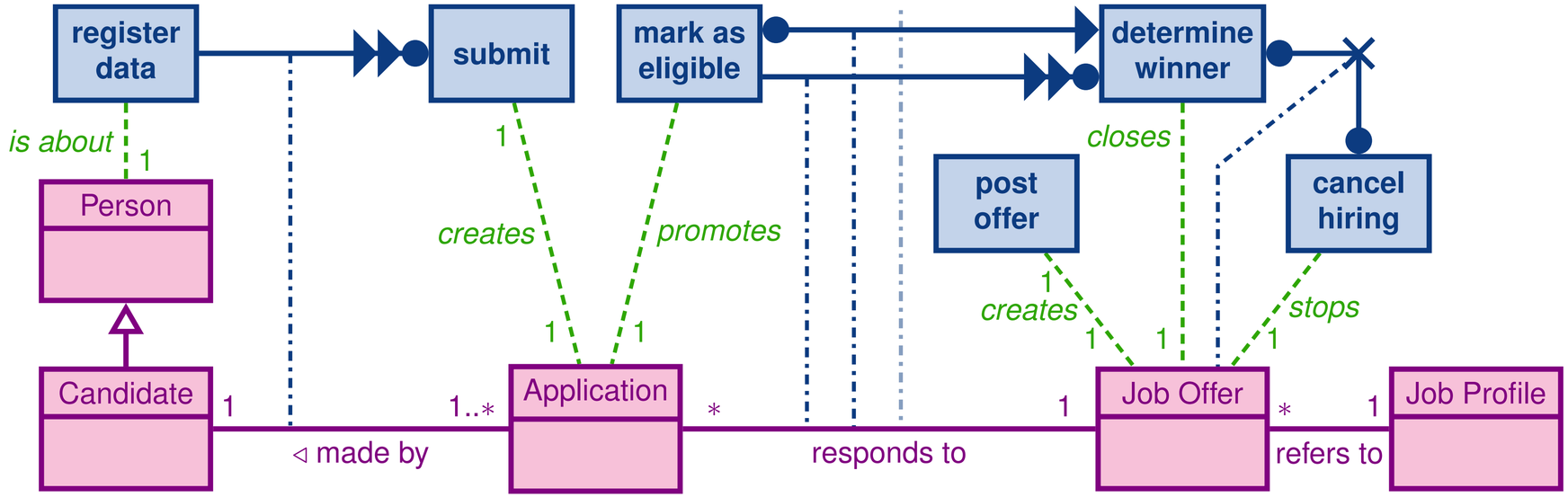
Object-centric behavioral constraints

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**.

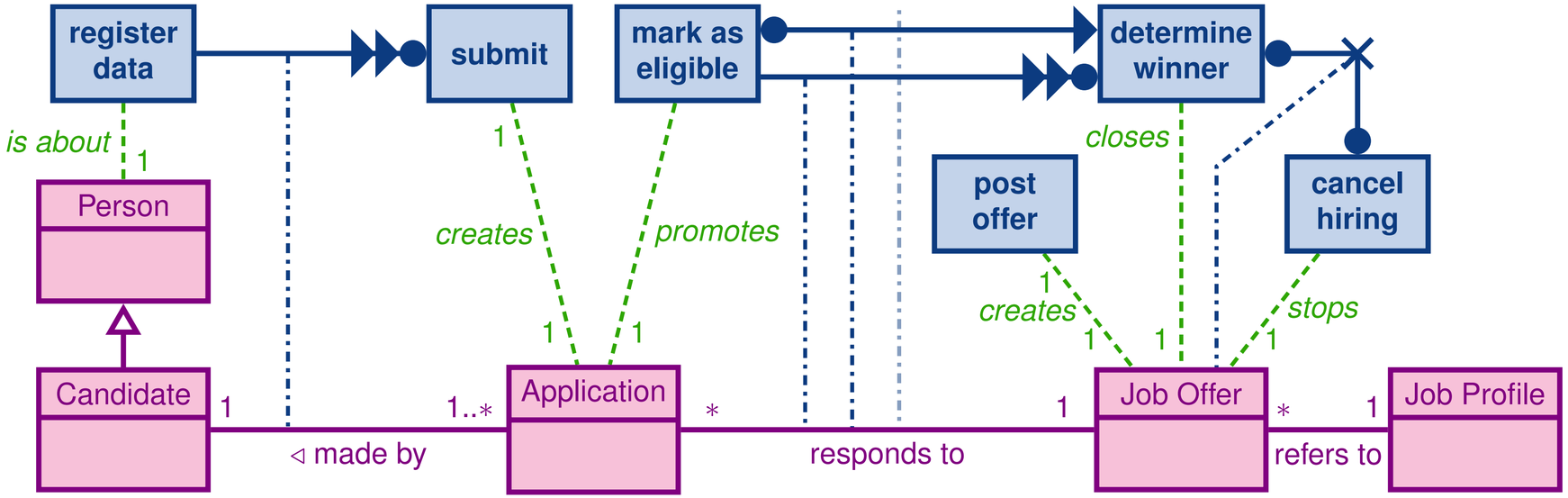


Object-centric behavioral constraints

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.

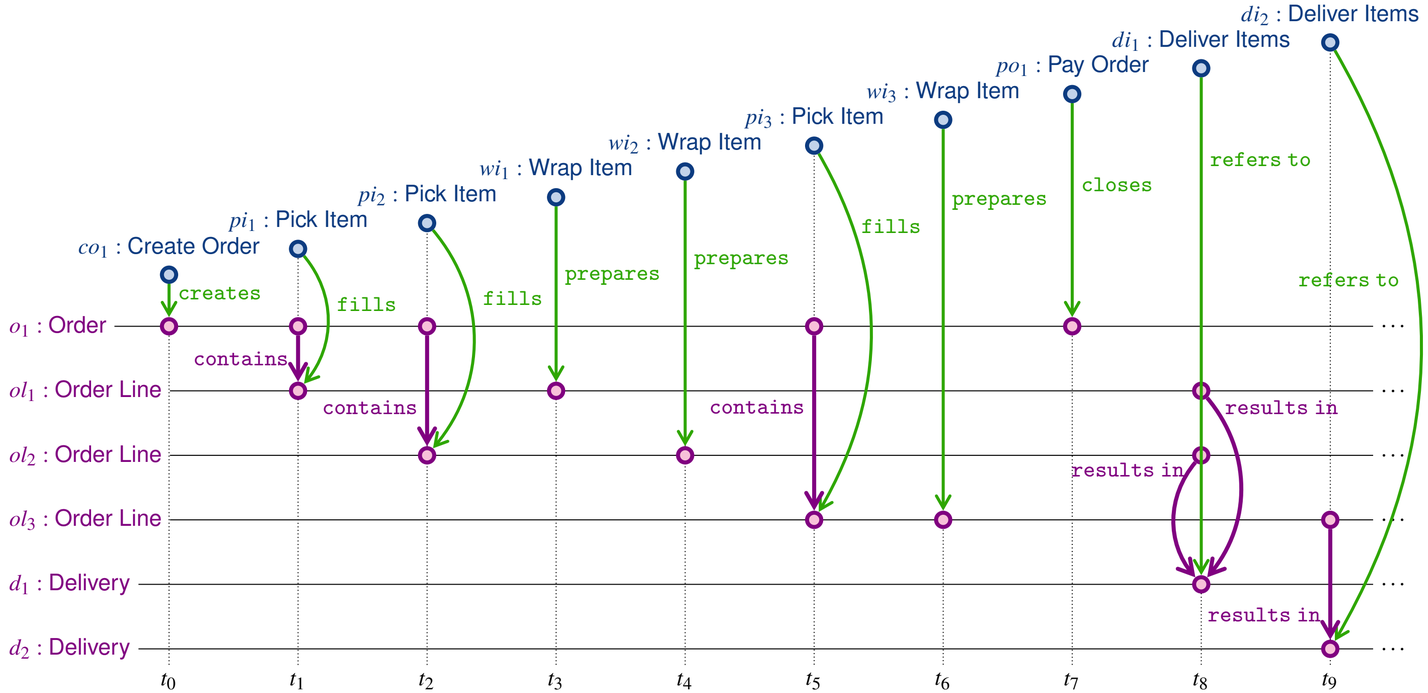


Semantics and formalization

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