Specifying a Usage Control System

SACMAT 2023

Ulrich Schöpp, Chuangjie Xu,
Amjad Ibrahim, Fathiyeh Faghih,
Theo Dimitrakos

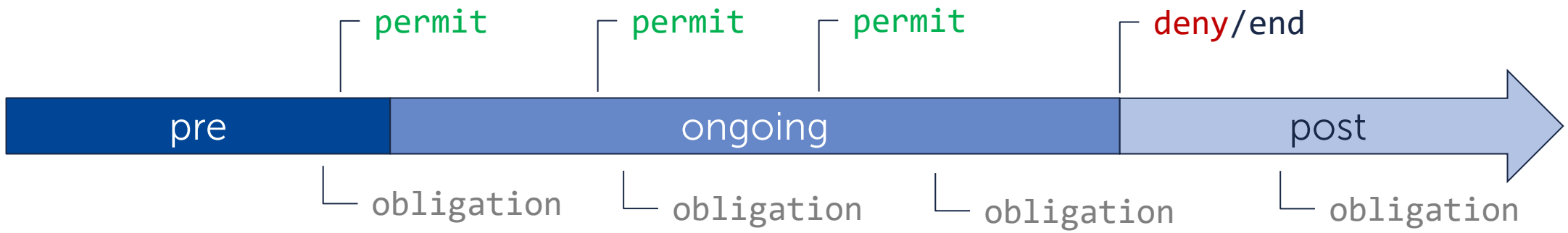
Attribute-Based Access Control

Policy standards like **XACML/ALFA** are **sufficient specifications** for **access control.**

ALFA policy

```
policy vm_actions {
  target clause Attributes.object == "Virtual Machine"

  apply firstApplicable

  rule start {
    target clause Attributes.action == "start"
                  and Attributes.user.role == "owner"
    condition Attributes.server_free_capacity > 100
    permit
  }
  …
```

Attribute-Based Usage Control



Usage control with **mutable attributes** is not sufficiently covered by existing specifications.

Authorization
- continuous re-evaluation during access/usage

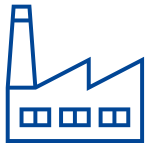
Obligations
- log access
- charge customer
- redact private data
- delete personal information

Conditions
- changing environment attributes

Attribute-Based Usage Control

Example: Passing usage policies as data

Hospital



Pharmaceutical Company

Name	DOB	Hemoglobin	...	Usage
John Doe	21/2/68	14	...	John's Policy
Jane Roe	03/11/03	15.1	...	Jane's Policy
...	...	...	...	...

John's data usage policy

```
policy data_usage {
  target clause …
  apply denyUnlessPermit

  rule pre {
    permit
    on permit {
      obligation { anonymizeData }
    }
  }
  …
```

Attribute-Based Usage Control

We need a **practical specification** for **implementations** of **usage control systems**.

- **Specification**

  Precise specification for policy semantics and usage control systems

- **Analysis**

  Understanding policies by automated analysis of usage control situations

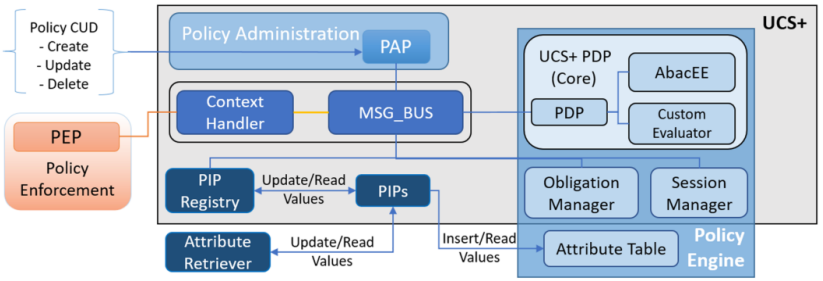
- **Verification**

  Testing and monitoring for usage control

Specifying Usage Control Systems

We define a specification for usage control systems.

UCS+ Usage Control System
[Dimitrakos et al., 2020]



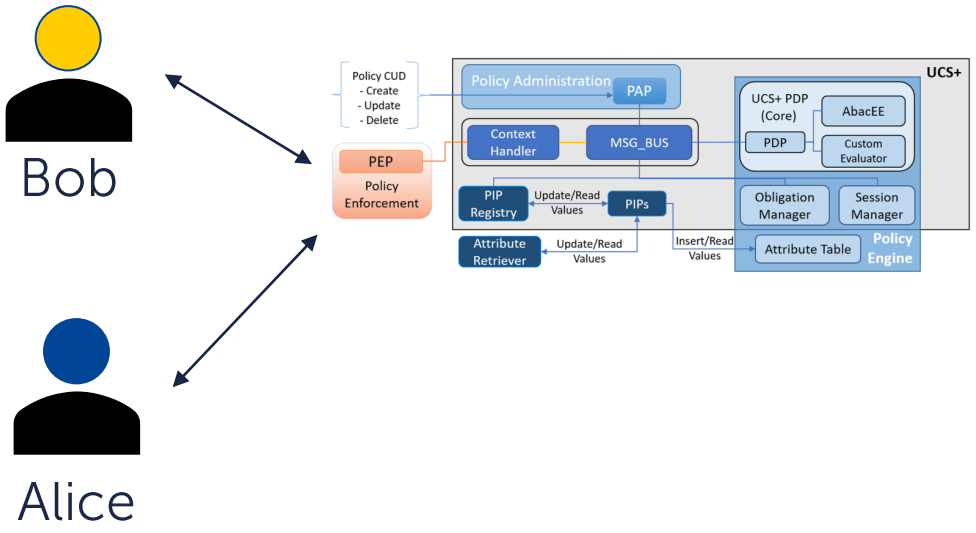
- C++ implementation
- Extension of UCON model [Park & Sandhu, 2004]

ALFA Policies

```
policy vm_usage {
  apply firstApplicable

  rule before_use {
    target clause Attributes.ucs.step.pre
    condition Attributes.accept_terms
    permit
  }
  rule during_use {
    target clause Attributes.ucs.step.ongoing
    condition Attributes.accept_terms && Attributes.credits > 0
    permit
  }
  rule after_use {
    target clause Attributes.ucs.step.post
    permit
    on permit { obligation sendInvoice }
  }
```

Specification Approach

The **system state** abstractly models a snapshot of mutable data and consequences.



State

PIP

Attributes
- today: Mo

Session 1

Attributes
- user: bob
- resource: X
- today: Mo

Phase
- pre

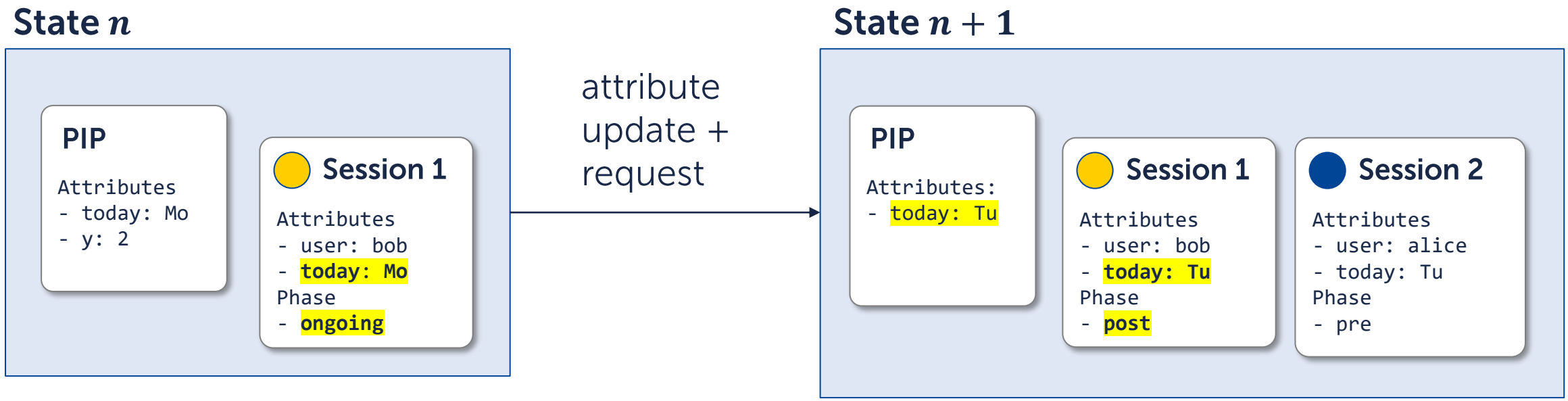
Session 2

Attributes
- user: alice
- resource: Y
- today: Mo

Phase
- ongoing

Specification Approach

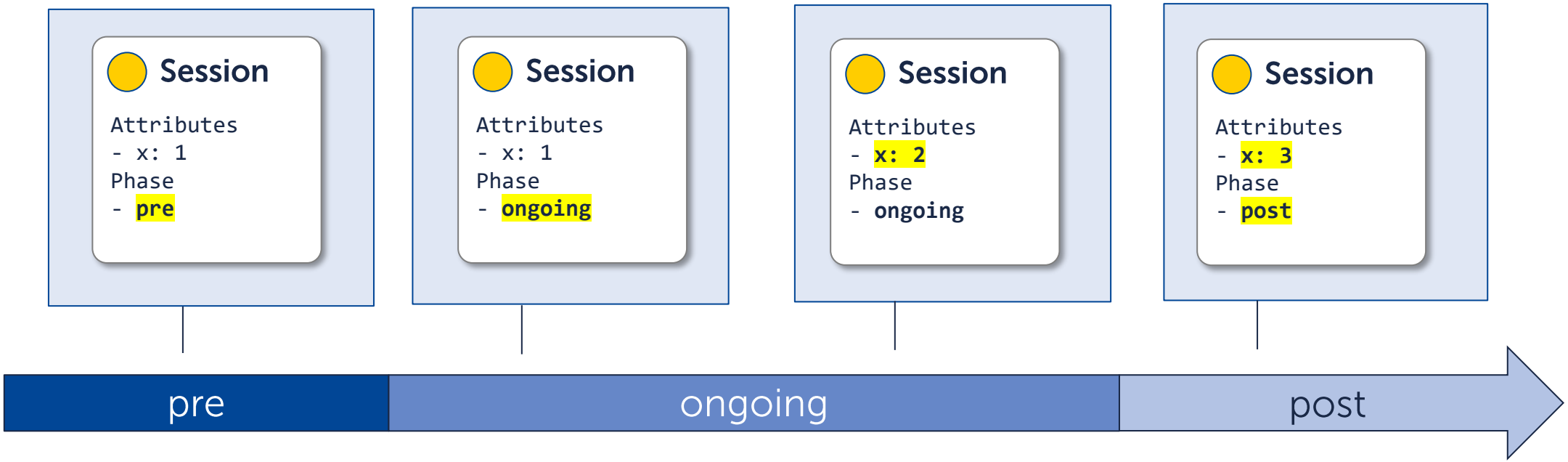
State transitions model change over time.



Specification Approach

Individual sessions implement continuous usage control

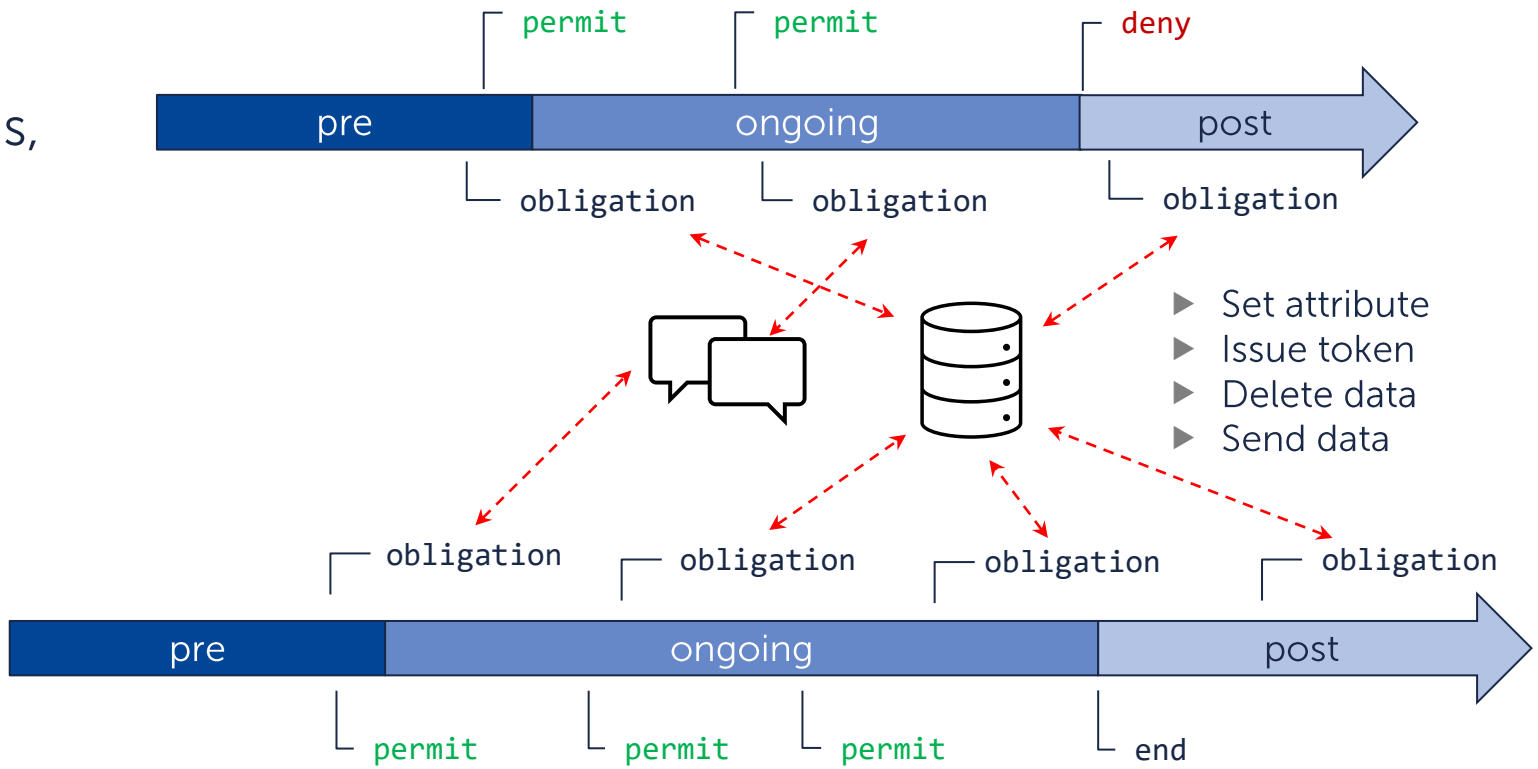
- Conceptually: reevaluate policy after any attribute change
- Non-trivial with concurrent sessions



Specification

Our specification accounts for

- policy decisions and obligations,
- multiple concurrent sessions,
- mutable attributes,
- obligation effects,
- synchronization of sessions and obligations,
- delegation,
- ...



There are several reasonable choices at many points.

Examples

Mutual exclusion [Park & Sandhu, 2004]
Limited number of simultaneous uses.

Visibility of updates
Policy writers may expect to see all updates.

```
policy visibiliy {
  apply denyOverrides

  rule denyIfUntrusted {
    condition Attributes.trustLevel < 50
    deny
    on deny {
      obligation requestReview
    }
  }
  …
```

Implementation in Answer Set Programming (ASP)

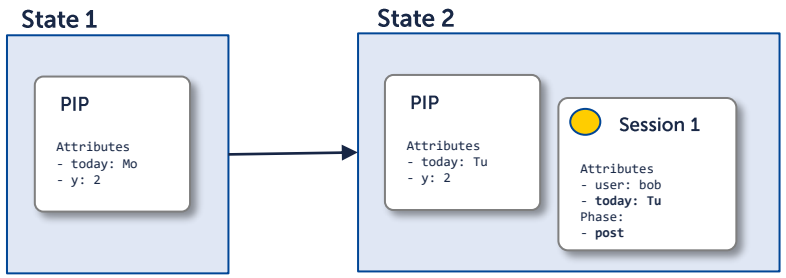
ALFA policy

```
policy smartOfficePolicy {
  target clause user == "scheduler"
           and device == "heating"
           and inhabited == true

  …
```

Propositional logic (ASP)

```
policy(smartOfficePolicy,root,firstApplicable).

match_target(smartOfficePolicy,Session,State) :-
  session_attribute(Session,"user","scheduler",State),
  session_attribute(Session,"device","heating",State),
  session_attribute(Session,"inhabited",true,State).
…
```

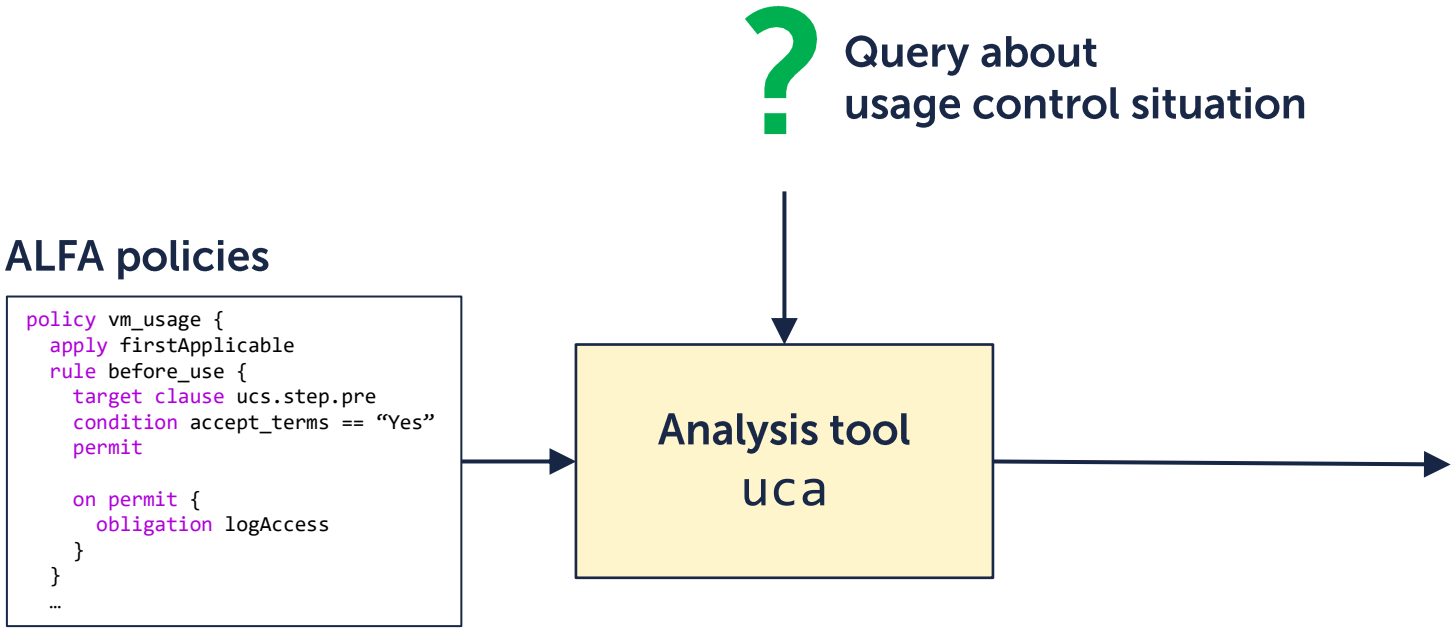
Specification of Usage Control

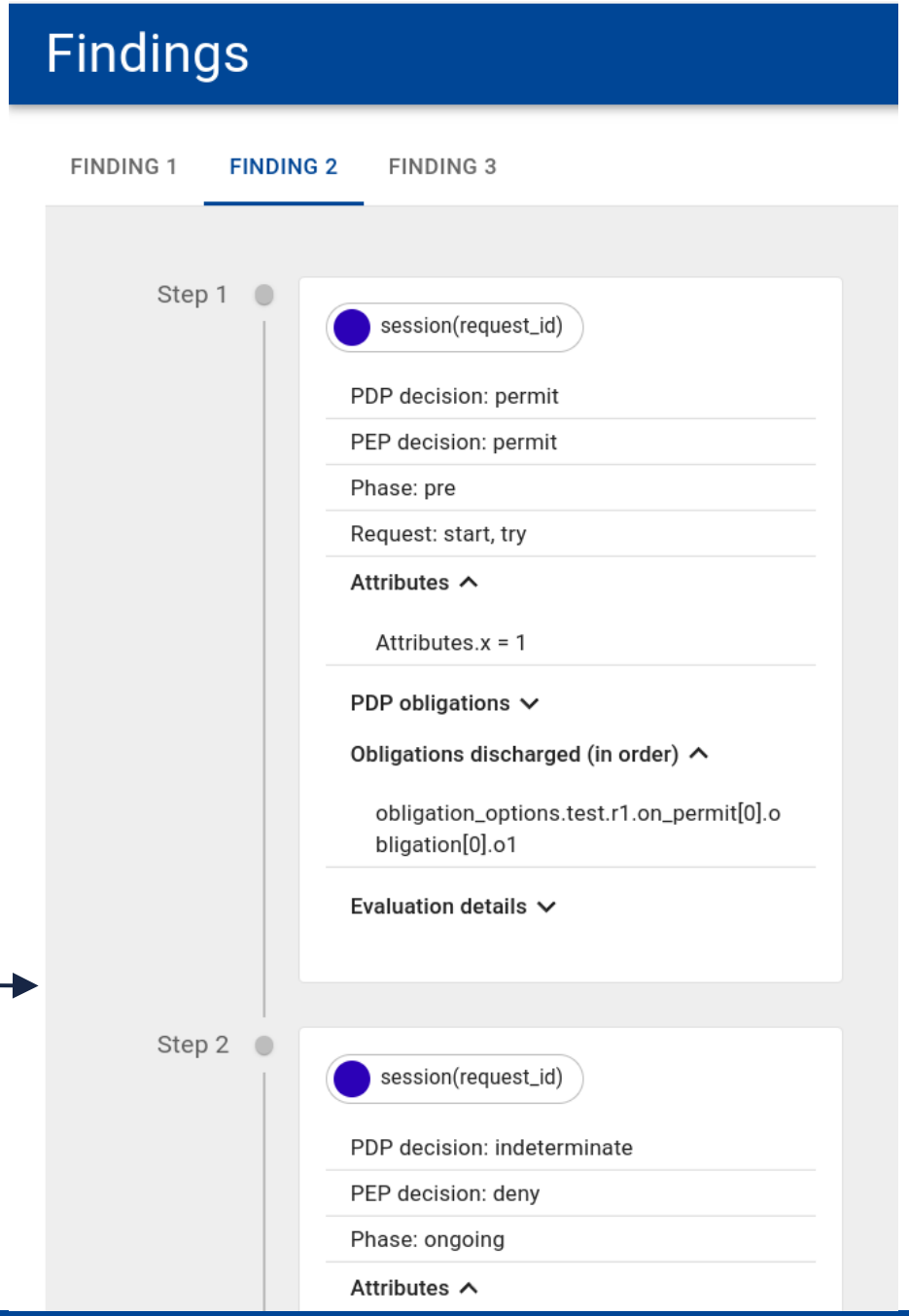



```
session_phase(Session, PhaseNext, T+1) :-
  session_phase(Session, Phase, T),
  phase_next(Session, PhaseNext, T),
  state(T+1).

phase_next(Session, "ongoing", T) :-
  session_phase(Session, "pre", T),
  session_pdp_decision(Session, permit, T),
  start_request(Session, T).
```

Application

Tool for automated analysis of usage control situations





Many thanks.

Any questions?

Contact

PD Dr. Ulrich Schöpp
Safety and Security

fortiss GmbH
Guerickestr. 25 • 80805 Munich • GERMANY
www.fortiss.org

contact@ulrichschoepp.de