Normative Programming

Jomi Fred Hübner

Universidade Federal de Santa Catarina
Departamento de Automação e Sistemas
http://jomi.das.ufsc.br

WESAAC 2015 @ Niterói RJ

Agenda

- Norms in MAS
- Programming Norms

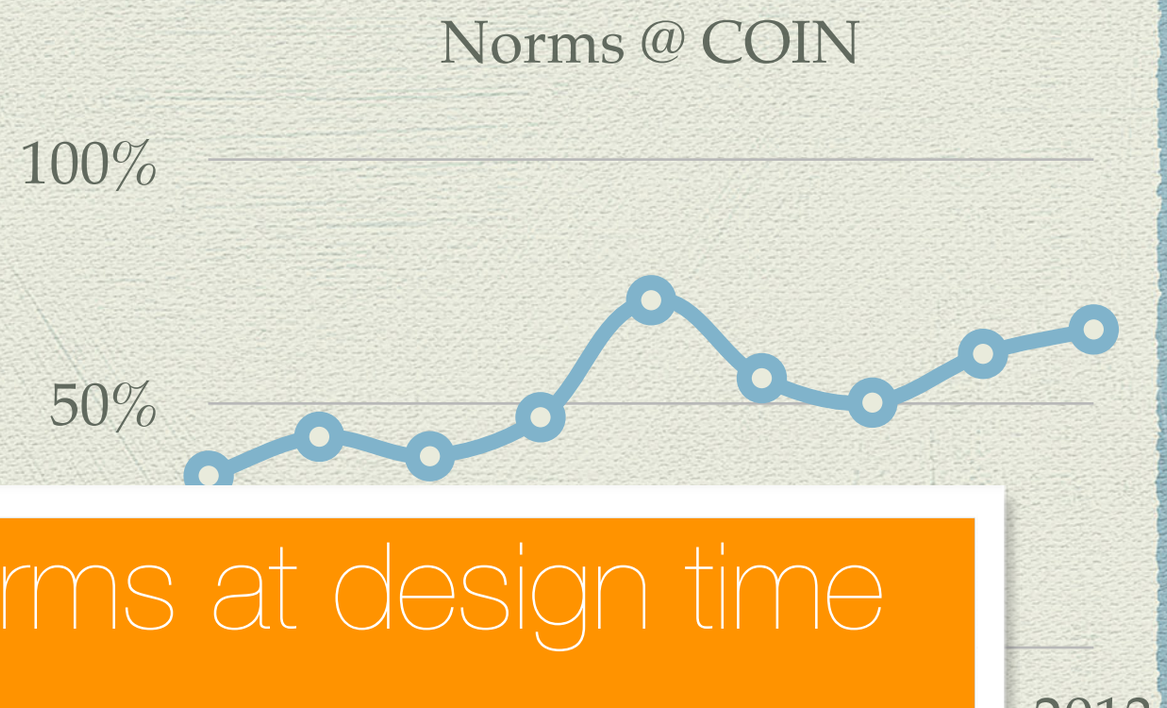
(regulative) Norms

- Expected (social) behavior
- Usual elements [Elinor Ostrom]
  - when a norm is applicable
  - to whom it applies
  - deontic operator
  - aim
  - [sanction]

when an auction is **finished**, the **bidder** is **obliged** to **pay** its offer, otherwise s/he will be **fined**

Norms in MAS

- Perspectives
  - AOSE (design time)
  - COIN (run time)
- Hot Topic
  - 52% (83/161) pap



norms at design time

- ✶ agents cannot reason about them
- ✶ norms cannot be changed at runtime

Why norms (at runtime)?

- To deal with open MAS and autonomous agents
  - "agents can enter or leave freely and neither the number, nor the behaviour, nor the way in which the agents interact and access shared resources can be known at design time" [Piunti]

Why norms (at runtime)?

- To program at a higher level
- To program the overall system and not a machine, a process, an object, or an agent
- The MAS specification does not need to be reduced to lower level concepts until it can be programmed (e.g. as processes)

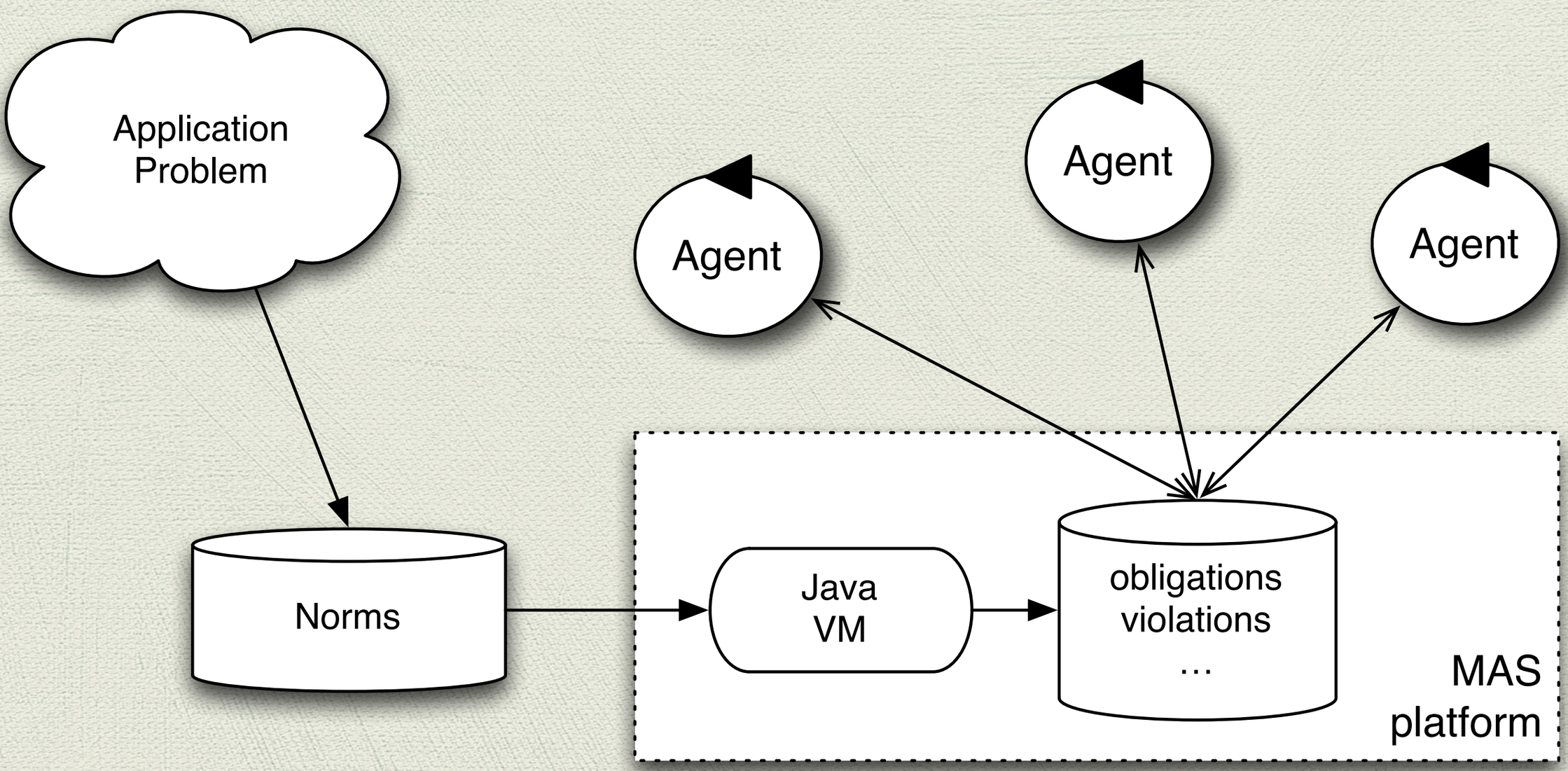
Agents level

- Norm aware reasoning
  - recognition
  - adoption
  - compliance
  - revision

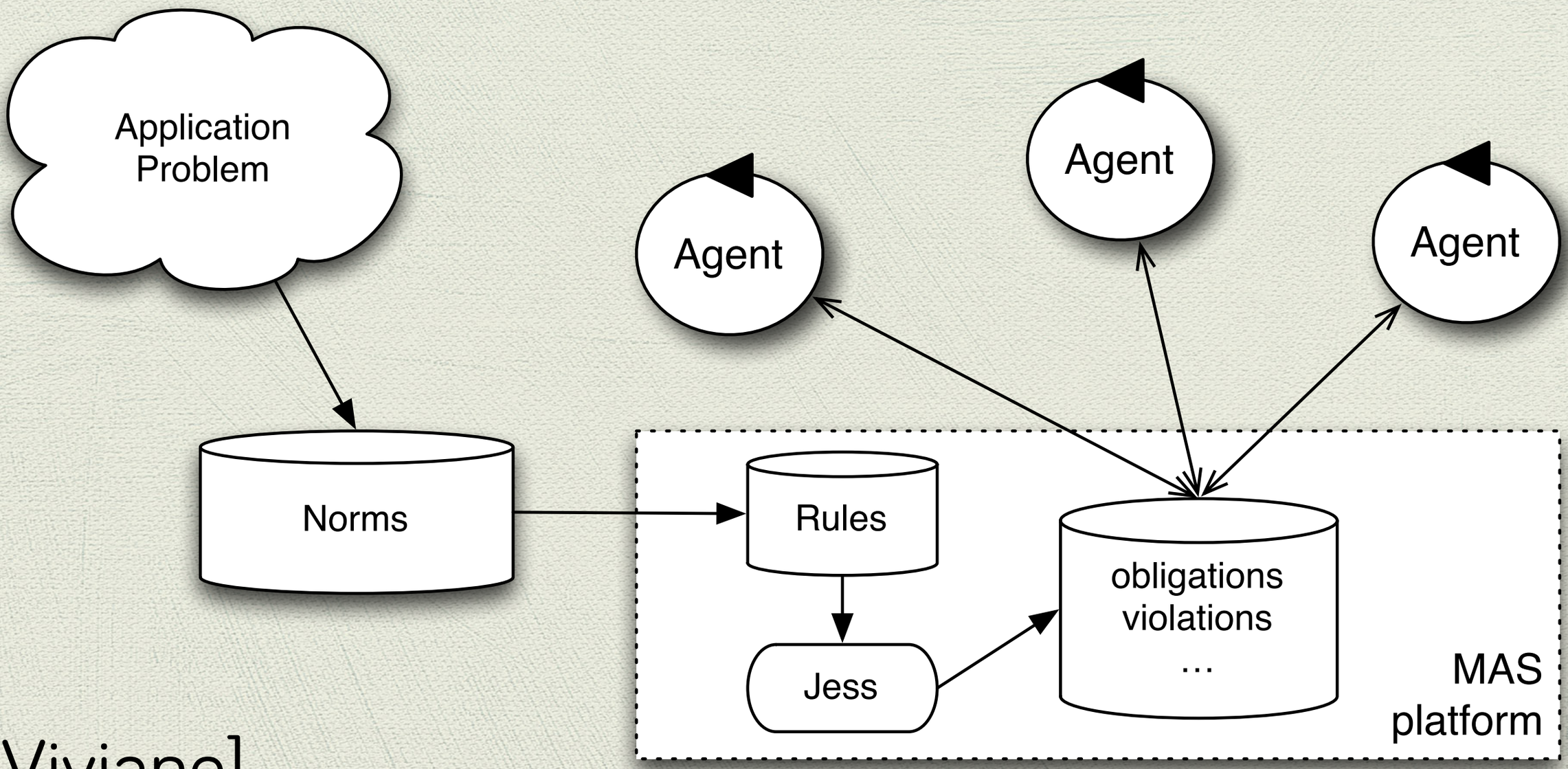
System level

- The system platform manages norms
  - independent from the agents
- Mechanisms
  - regimentation
  - enforcement (detection, sanctions, reputation, ...)

Normative Platform



Normative Platform



[Viviane]

Normative pr

programming an MAS is not programming (only) agents

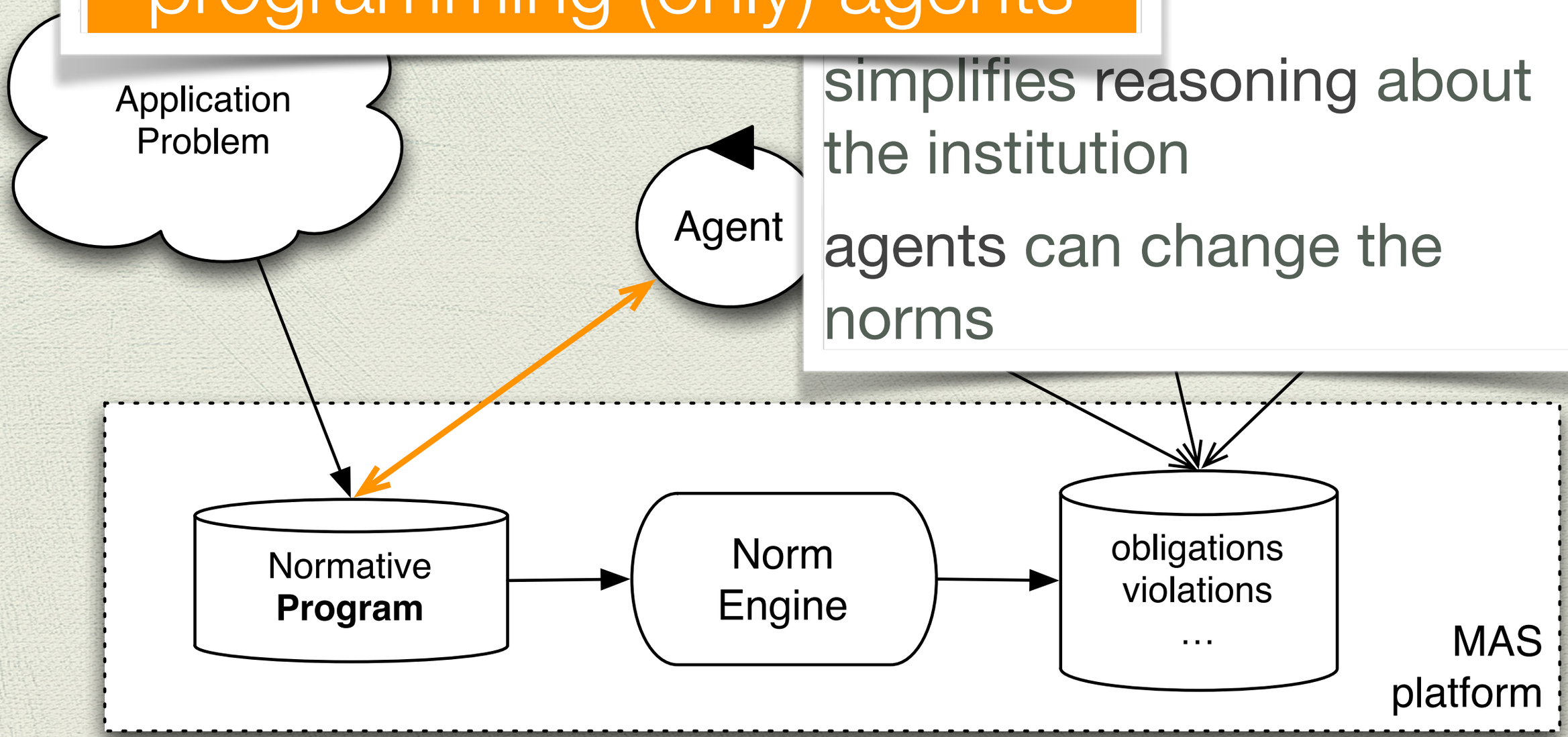
objectives

control [malicious] agents

to achieve
] goals

simplifies reasoning about the institution

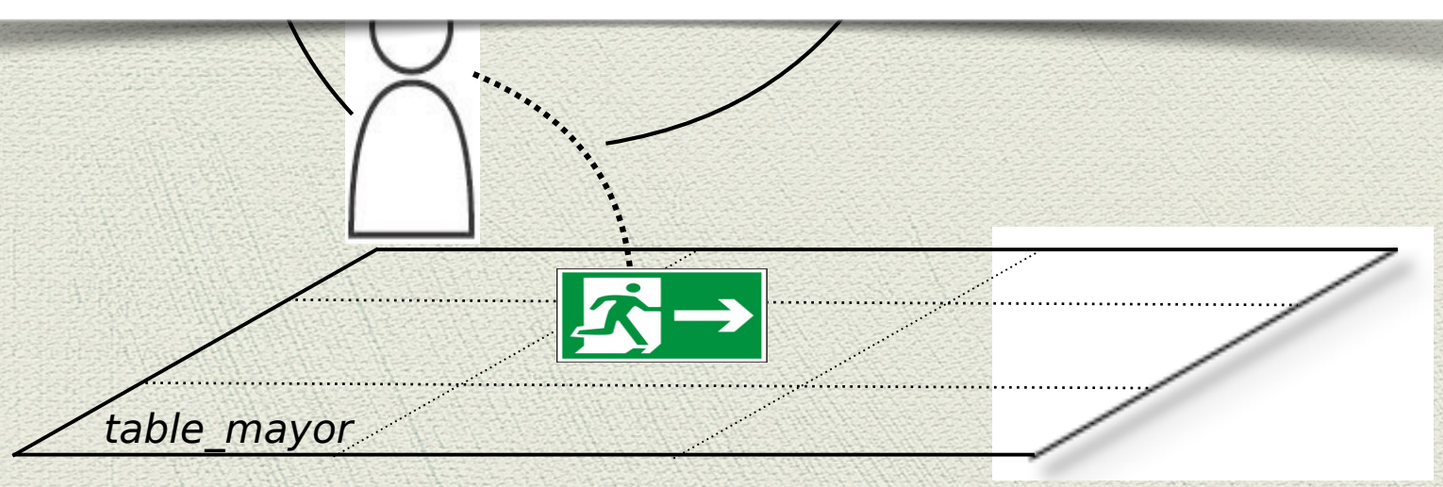
agents can change the norms



Example:
Situated Artificial Institution (SAI)

The **mayor** is obliged to **command evacuation**

The **mayor** is obliged to command **evacuation** the **area is risky**



Norms

Status

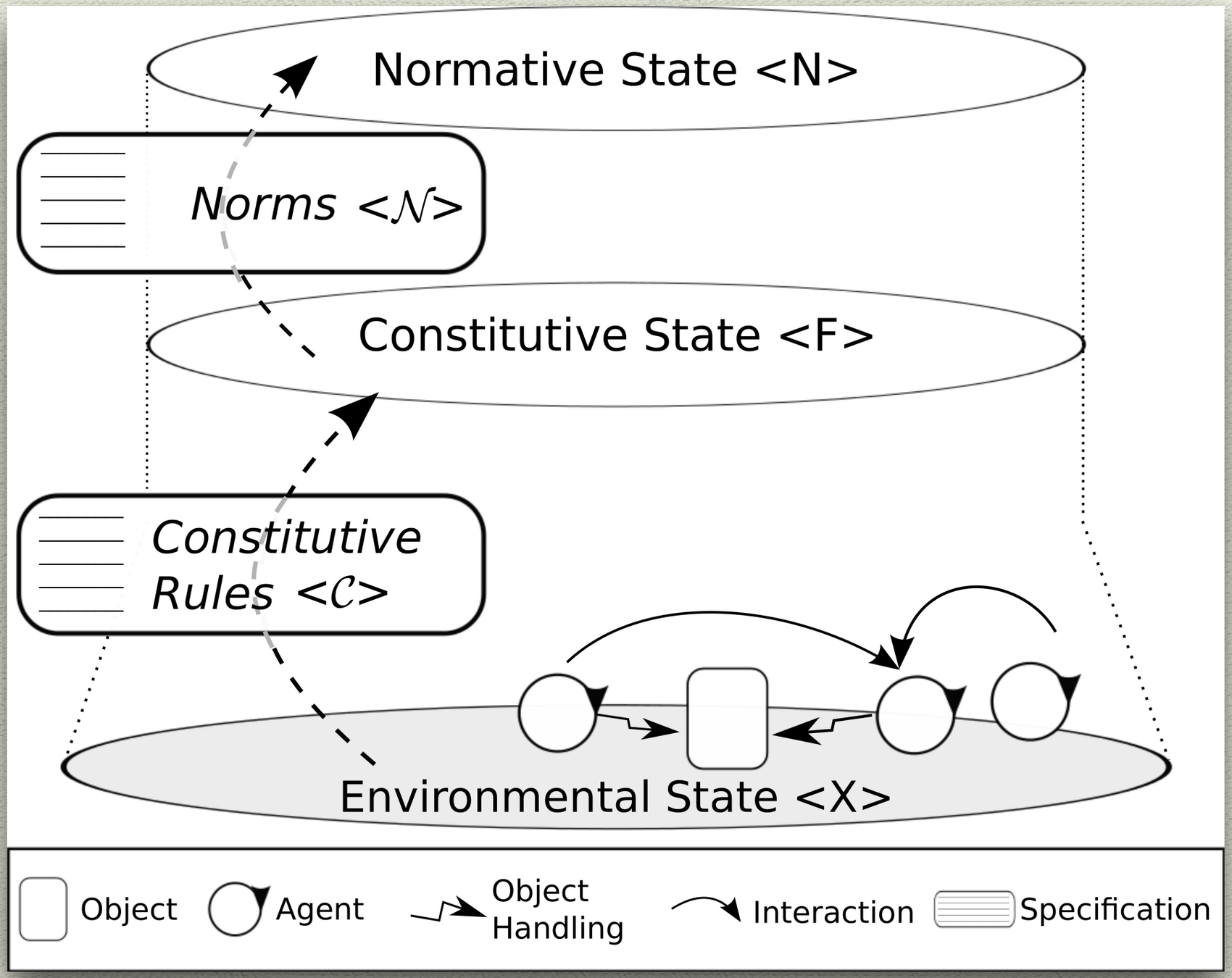
Functions

Constitutive Rules

Environment

[Brito]

Example:
Situated Artificial Institution (SAI)



[Brito]

Example: 2OPL

Example (Train Station)

```
Facts:
    { -at_platform , -in_train , -ticket }

Effects:
    { -at_platform }    enter           { at_platform },
    { -ticket }         buy_ticket      { ticket },
    { at_platform , -in_train }
                        embark
                            { -at_platform, in_train }
Counts_as rules:
    { at_platform , -ticket } => { viol_ticket },
    { in_train , -ticket }    => { viol_|_ }

Sanction_rules:
    { viol_ticket } => { fined_10 }
```

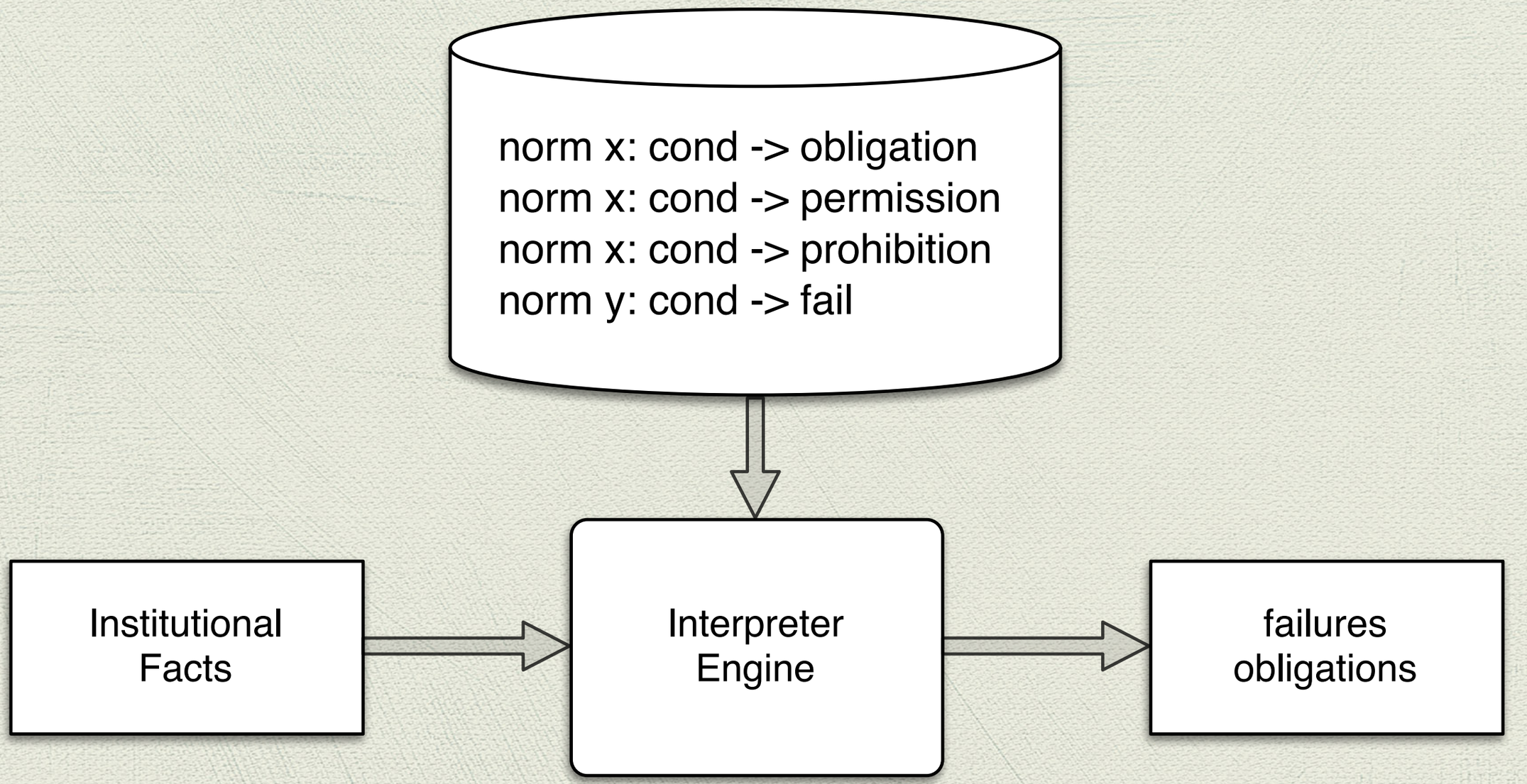
[Mehdi]

Example: NPL

when an auction is **finished**, the **bidder** is **obliged** to **pay** its offer, otherwise s/he will be **fined**

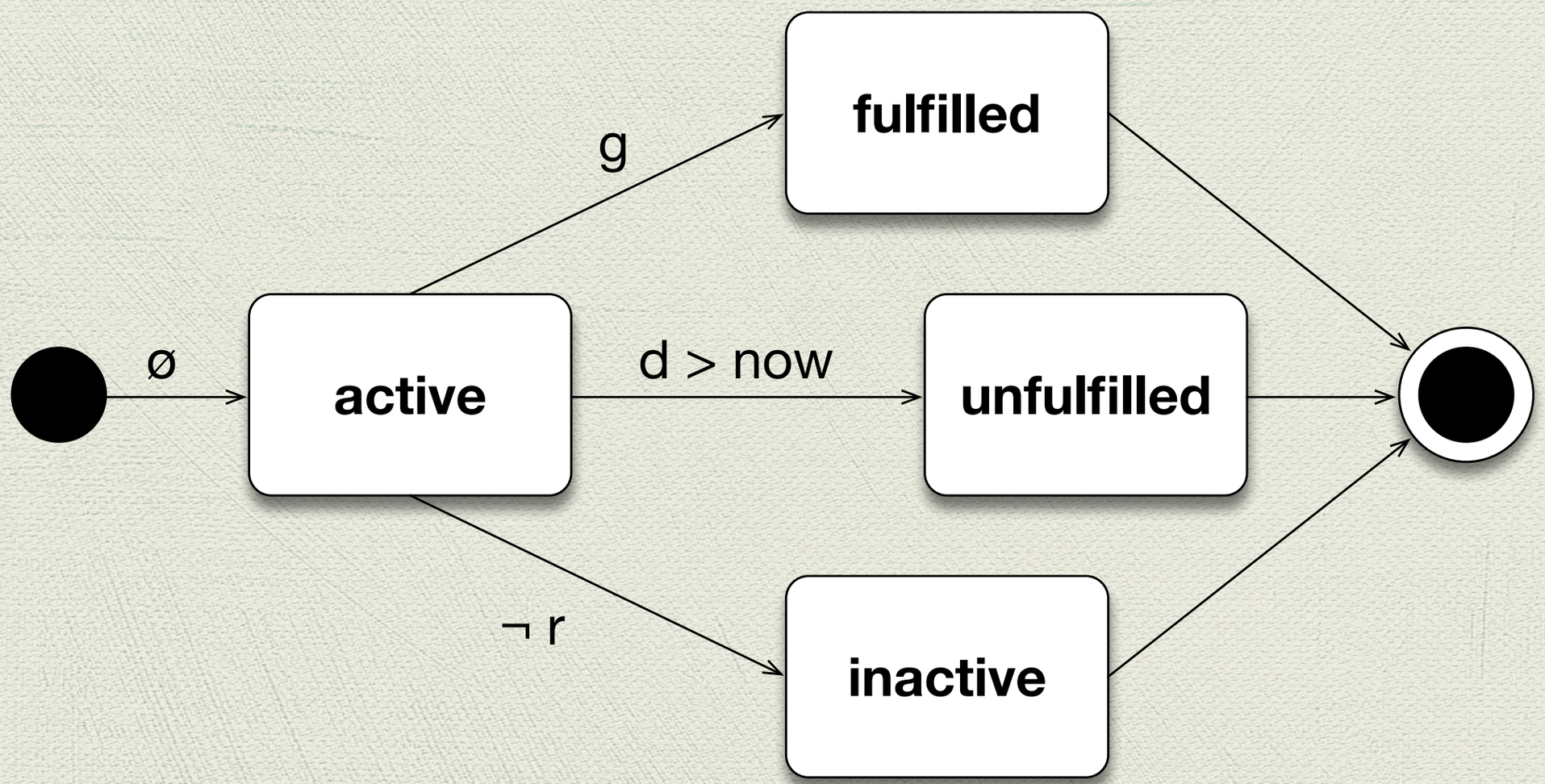
- norm auction_pay:
  finished(Auction) &
  play(Ag,bidder,Auction)
  -> obligation(Ag,
  winner(Ag,Auction),
  paid(Auction),
  ‘now + 2 days‘)

NPL Interpreter



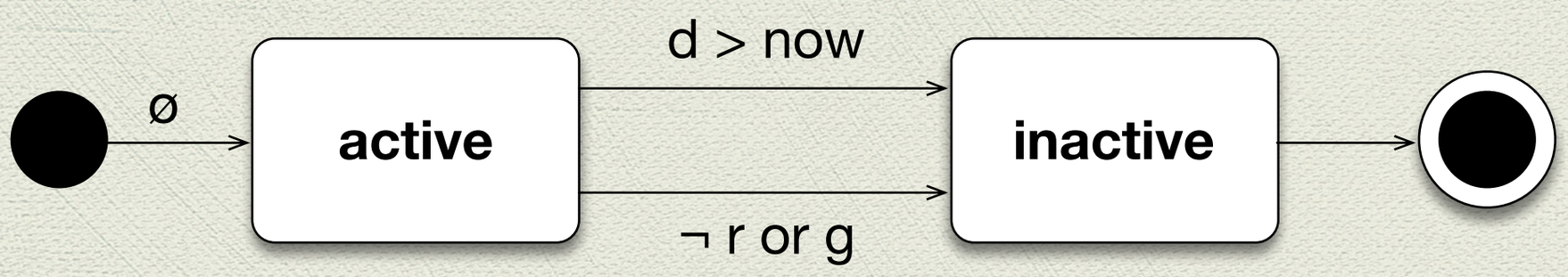
Obligations in NPL

- norm n: ⊘ -> obligation(a,r,g,d)



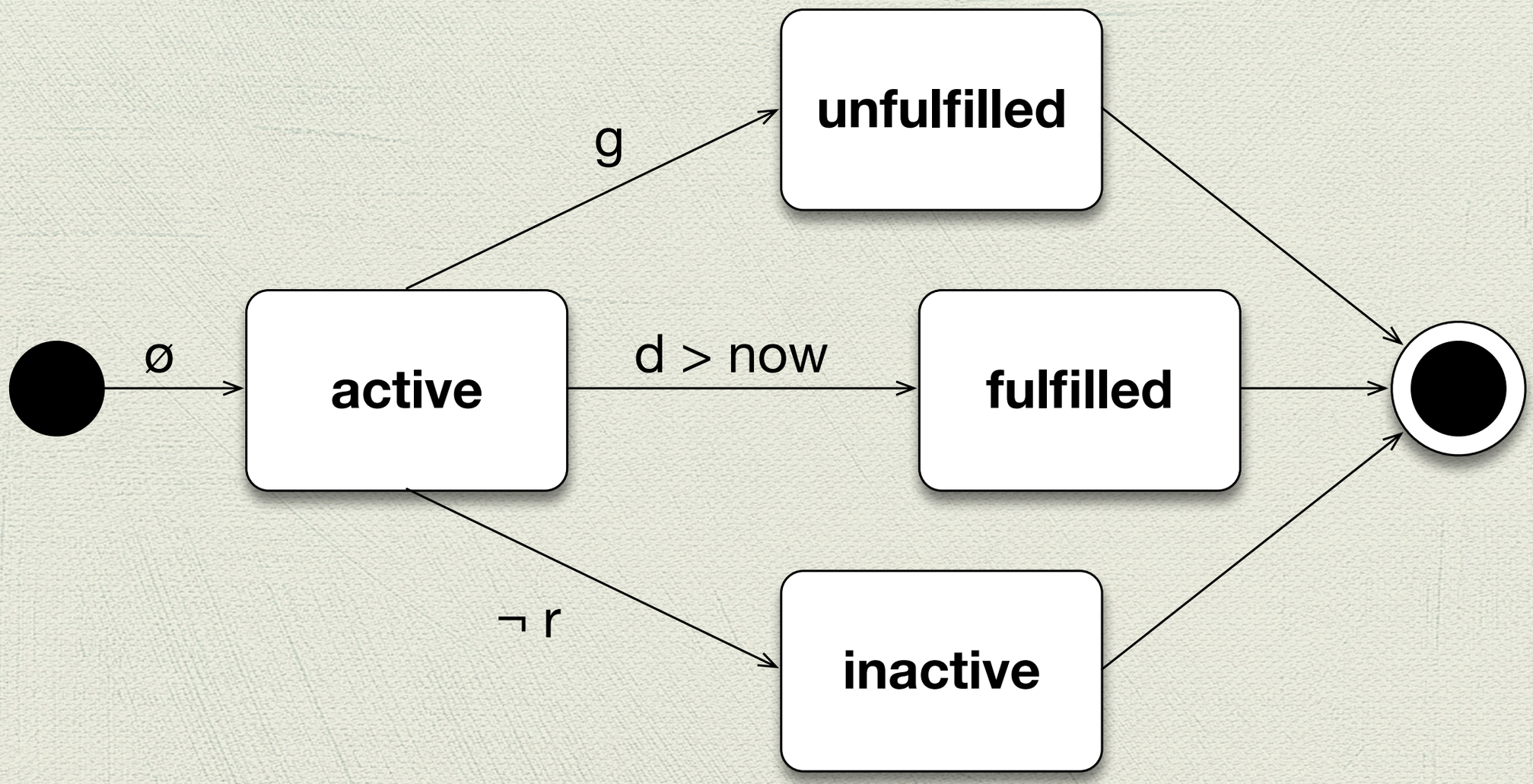
Permissions in NPL

- norm n: ⊘ -> permission(a,r,g,d)

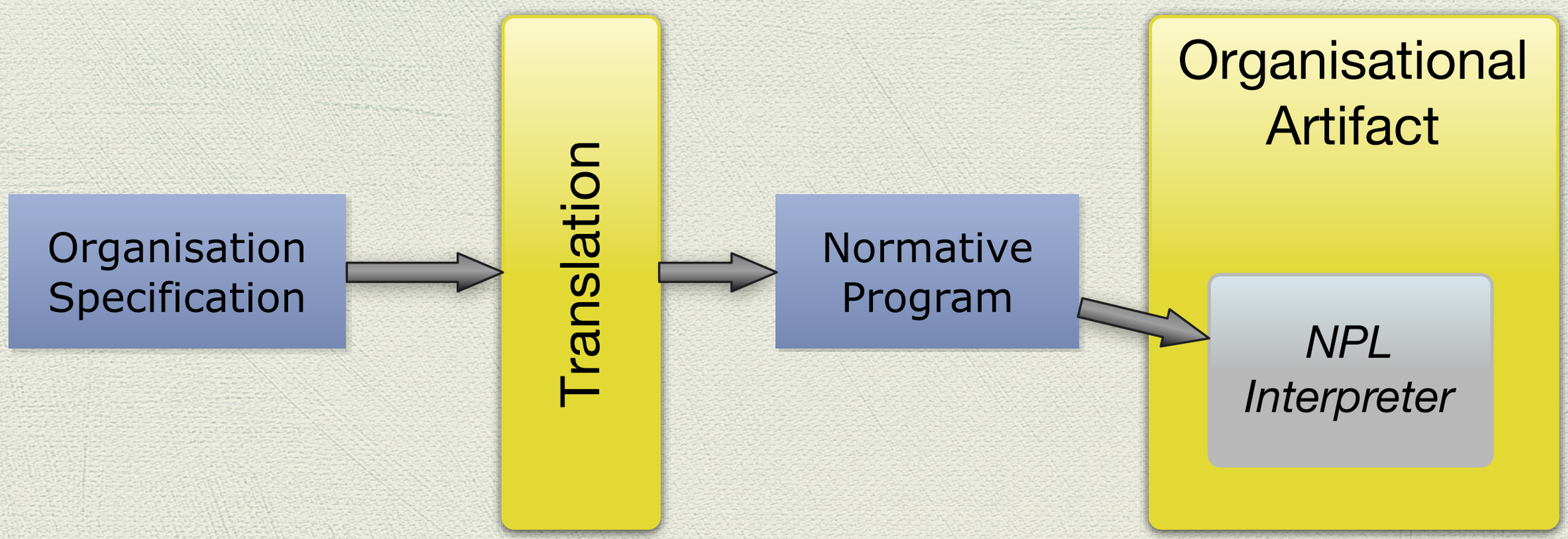


Prohibition in NPL

- norm n: ⊘ -> prohibition(a,r,g,d)



Example: NOPL



Example: NOPL

- norm ngoa:

  committed(A,M,S) &

  goal(M,G, achievement, D) &

  well_formed(S)

  -> obligation(A,

  enabled(G,S),

  achieved(S,G),

  ‘now‘ + D)

Example: NOPL

- norm ngoa:

  committed(A,M,S) &

  goal(M,G, performance, D) &

  well_formed(S)

  -> obligation(A,

  enabled(G,S),

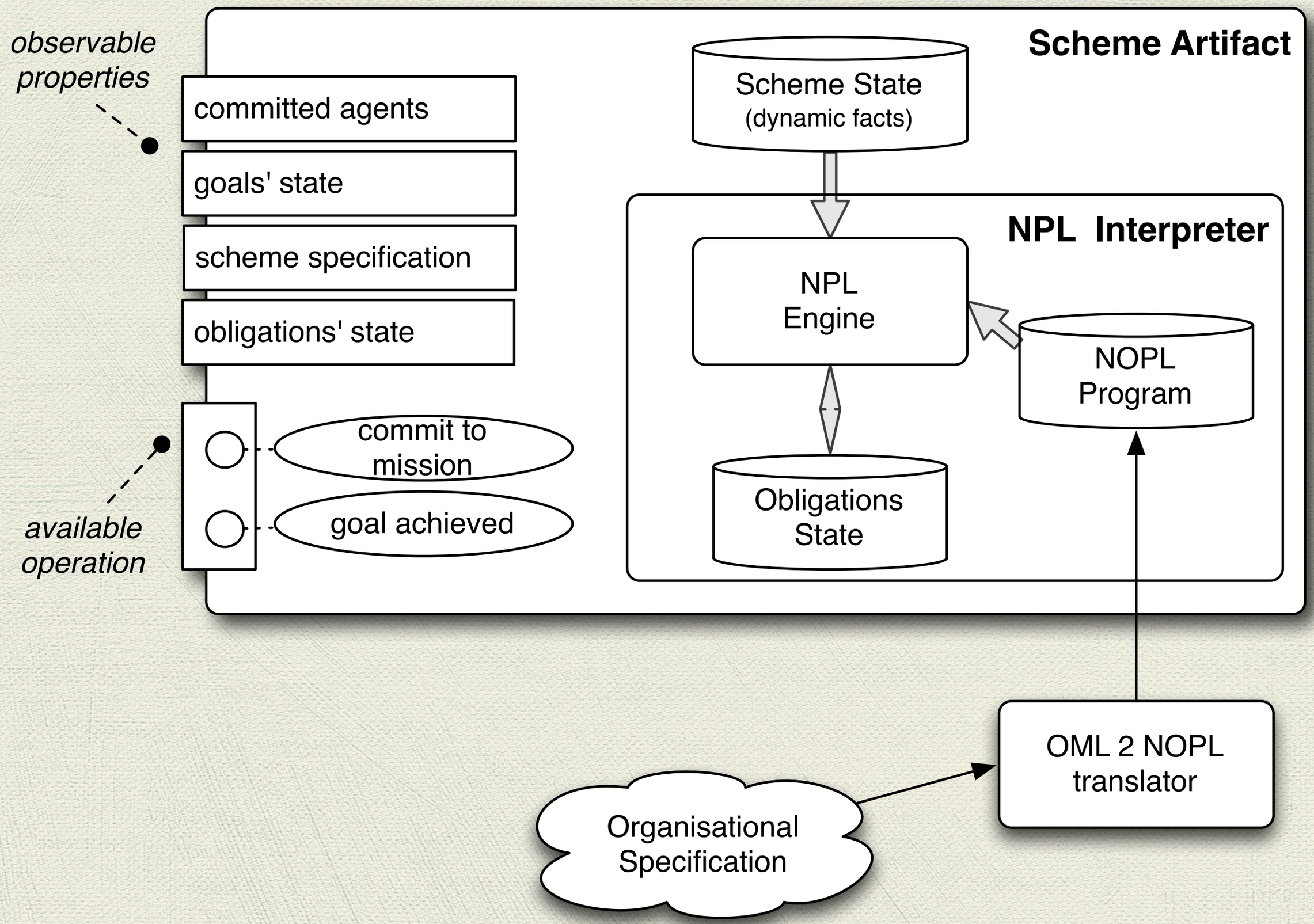
  done(S,G,A),

  ‘now‘ + D)

Example: NOPL

- norm mission_cardinality:
  scheme_mission(M,_,MMax) &
  mplayers(M,S,MP) & MP > MMax
  -> fail(mission_cardinality).

- norm mission_cardinality:
  scheme_mission(M,_,MMax) &
  mplayers(M,S,MP) & MP < MMax
  responsible(Gr,S)
  -> obligation(A, plays(A,editor,Gr),
  committed(A,ms,_), ‘now‘+‘1 hour‘).


observable properties
committed agents
goals' state
scheme specification
obligations' state
commit to mission
goal achieved
available operation
Scheme Artifact
Scheme State
(dynamic facts)
NPL Interpreter
NPL
Engine
NOPL
Program
Obligations
State
OML 2 NOPL
translator
Organisational
Specification

Alg. for org actions

1: *oe* is the state of the organisation managed by the artifact
2: *p* is the current NOPL program
3: *npi* is the NPL interpreter
4: **when** an operation *o* is triggered by agent *a* **do**
5: $oe' \leftarrow oe$ // creates a "backup" of the current *oe*
6: executes operation *o* to change *oe*
7: $f \leftarrow$ a list of predicates representing *oe*
8: $r \leftarrow npi(p, f)$ // runs the interpreter for the new state
9: **if** $r =$ fail **then**
10: $oe \leftarrow oe'$ // restore the backup state
11: **return** fail operation *o*
12: **else**
13: update obligations in the observable properties
14: **return** succeed operation *o*

..:: ORA4MAS Artifacts GUI ::..

... Scheme Board bhsch (build_house_sch) ...

organisation entity | normative state | normative facts | normative program | specification

Normative State: *scheme(build_house_sch)*

state	agent	reason (norm)	goal
active	companyD12	ngoal("bhsch",prepare_site,site_prepared)	achieved("bhsch",site_prepared,companyD12)
unfulfilled	companyD12	ngoal("bhsch",prepare_site,site_prepared)	achieved("bhsch",site_prepared,companyD12)

bhsch_group

bhsch

History

```
fulfilled:   obligation(companyC1,n10,committed(companyC1,paint_house,"bhsch"),1314377299506)[create
fulfilled:   obligation(companyA,n8,committed(companyA,install_plumbing,"bhsch"),1314377299549)[crea
fulfilled:   obligation(companyC1,n9,committed(companyC1,install_electrical_system,"bhsch"),1314377
fulfilled:   obligation(companyD8,n7,committed(companyD8,fit_doors,"bhsch"),1314377299550)[created(
fulfilled:   obligation(companyD8,n6,committed(companyD8,fit_windows,"bhsch"),1314377299551)[create
created:     obligation(companyD12,ngoal("bhsch",prepare_site,site_prepared),achieved("bhsch",site_
created:     obligation(companyE,ngoal("bhsch",l
fulfilled:   obligation(companyE,ngoal("bhsch",l
unfulfilled: obligation(companyD12,ngoal("bhsch"
```

http://moise.sf.net

What is the (best) language to program the institutional platform?

- java?
- rules?
- norms?
  - translation instead of coding
  - agent can reason on norms or institutional specification (both are available)

ORA4MAS

- ✳ 80% code NOPL (obligations)
- ✳ 10% code CArtAgO (agent interface)
- ✳ 10% code Java

Open issues

- monitoring “big brother”
  - how to get all data
  - how to deal with so much data on time
- regimentations or sanction
  - how to chose the best strategy
- integration with constitutive rules
- normative “modules”

Summary

- Normative Programming
  - few code with a lot of meaning
  - at runtime
  - for agents to reason about the institution
  - for designers (and agents) to specify institutions