# Representing and reasoning about policy for agent-based simulation

Julian Padget

Department of Computer Science, University of Bath

April 2, 2019

## Contents

## Contents

## Unpacking the title

- normative models for intelligent agents
- agent architectures for normative reasoning
- applications in social simulation, security, games, legal reasoning, software engineering, data analytics
- norms ≡ policies ≡ regulations ≡ narratives ≡ requirements
- current work on:
  - normative reasoning as a service (Padget et al. 2018)
  - semantic representation of policy (on-going)
  - socio-cognitive technical systems (SCTS) (Noriega et al. 2017)
  - policy-making as an instance of SCTS (on-going)
  - use in social simulation (why I'm here)
- context from previous work
  - Institutional Action Language: InstAL (Padget et al. 2016b)
  - NJason: extension of Jason agent platform (Lee et al. 2014)

## Unpacking the title

- normative models for intelligent agents
- agent architectures for normative reasoning
- applications in social simulation, security, games, legal reasoning, software engineering, data analytics
- norms ≡ policies ≡ regulations ≡ narratives ≡ requirements
- current work on:
  - normative reasoning as a service (Padget et al. 2018)
  - semantic representation of policy (on-going)
  - socio-cognitive technical systems (SCTS) (Noriega et al. 2017)
  - policy-making as an instance of SCTS (on-going)
  - use in social simulation (why I'm here)
- context from previous work
  - Institutional Action Language: InstAL (Padget et al. 2016b)
  - NJason: extension of Jason agent platform (Lee et al. 2014)

## Unpacking the title

- normative models for intelligent agents
- agent architectures for normative reasoning
- applications in social simulation, security, games, legal reasoning, software engineering, data analytics
- norms ≡ policies ≡ regulations ≡ narratives ≡ requirements
- current work on:
  - normative reasoning as a service (Padget et al. 2018)
  - semantic representation of policy (on-going)
  - socio-cognitive technical systems (SCTS) (Noriega et al. 2017)
  - policy-making as an instance of SCTS (on-going)
  - use in social simulation (why I'm here)
- context from previous work
  - Institutional Action Language: InstAL (Padget et al. 2016b)
  - NJason: extension of Jason agent platform (Lee et al. 2014)

## Unpacking the title

- normative models for intelligent agents
- agent architectures for normative reasoning
- applications in social simulation, security, games, legal reasoning, software engineering, data analytics
- norms ≡ policies ≡ regulations ≡ narratives ≡ requirements
- current work on:
  - normative reasoning as a service (Padget et al. 2018)
  - semantic representation of policy (on-going)
  - socio-cognitive technical systems (SCTS) (Noriega et al. 2017)
  - policy-making as an instance of SCTS (on-going)
  - use in social simulation (why I'm here)
- context from previous work
  - Institutional Action Language: InstAL (Padget et al. 2016b)
  - NJason: extension of Jason agent platform (Lee et al. 2014)

## Unpacking the title

- normative models for intelligent agents
- agent architectures for normative reasoning
- applications in social simulation, security, games, legal reasoning, software engineering, data analytics
- norms ≡ policies ≡ regulations ≡ narratives ≡ requirements
- current work on:
  - normative reasoning as a service (Padget et al. 2018)
  - semantic representation of policy (on-going)
  - socio-cognitive technical systems (SCTS) (Noriega et al. 2017)
  - policy-making as an instance of SCTS (on-going)
  - use in social simulation (why I'm here)
- context from previous work
  - Institutional Action Language: InstAL (Padget et al. 2016b)
  - NJason: extension of Jason agent platform (Lee et al. 2014)

## Unpacking the title

- normative models for intelligent agents
- agent architectures for normative reasoning
- applications in social simulation, security, games, legal reasoning, software engineering, data analytics
- norms ≡ policies ≡ regulations ≡ narratives ≡ requirements
- current work on:
  - normative reasoning as a service (Padget et al. 2018)
  - semantic representation of policy (on-going)
  - socio-cognitive technical systems (SCTS) (Noriega et al. 2017)
  - policy-making as an instance of SCTS (on-going)
  - use in social simulation (why I'm here)
- context from previous work
  - Institutional Action Language: InstAL (Padget et al. 2016b)
  - NJason: extension of Jason agent platform (Lee et al. 2014)

## Why model policies?

- evidence-based policy-making
- safer AI
- explainable AI
- human accountability + responsibility in S(C)TS
- confidence in outcomes
- nothing new here? depends on
  - the policy-modelling language
  - who/what does the reasoning

## Why model policies?

- evidence-based policy-making
- safer AI
- explainable AI
- human accountability + responsibility in S(C)TS
- confidence in outcomes
- nothing new here? depends on
  - the policy-modelling language
  - who/what does the reasoning

## Why model policies?

- evidence-based policy-making
- safer AI
- explainable AI
- human accountability + responsibility in S(C)TS
- confidence in outcomes
- nothing new here? depends on
  - the policy-modelling language
  - who/what does the reasoning

## Why model policies?

- evidence-based policy-making
- safer AI
- explainable AI
- human accountability + responsibility in S(C)TS
- confidence in outcomes
- nothing new here? depends on
  - the policy-modelling language
  - who/what does the reasoning

## Why model policies?

- evidence-based policy-making
- safer AI
- explainable AI
- human accountability + responsibility in S(C)TS
- confidence in outcomes
- nothing new here? depends on
  - the policy-modelling language
  - who/what does the reasoning

## Why model policies?

- evidence-based policy-making
- safer AI
- explainable AI
- human accountability + responsibility in S(C)TS
- confidence in outcomes
- nothing new here? depends on
  - the policy-modelling language
  - who/what does the reasoning

## Why agent-based models?

- abstraction (equational) vs. synthesis (agent-based)
- ...or top-down vs. bottom-up
- approaches emphasize different dimensions

| | | |
|---|---|---|
| accuracy | granularity | fidelity |
| heterogeneity | precision | scalability |

- does a more complex model help understanding of complex systems?
- nothing new here? depends on
  - ratio of reactive : deliberative : generative behaviour
  - adaptation: hard-coded vs. data-driven behaviour
- policy is late-binding for simulation
- simulation is test environment for policy

## Why agent-based models?

- abstraction (equational) vs. synthesis (agent-based)
- ...or top-down vs. bottom-up
- approaches emphasize different dimensions

| | | |
|---|---|---|
| accuracy | granularity | fidelity |
| heterogeneity | precision | scalability |

- does a more complex model help understanding of complex systems?
- nothing new here? depends on
  - ratio of reactive : deliberative : generative behaviour
  - adaptation: hard-coded vs. data-driven behaviour
- policy is late-binding for simulation
- simulation is test environment for policy

## Why agent-based models?

- abstraction (equational) vs. synthesis (agent-based)
- ...or top-down vs. bottom-up
- approaches emphasize different dimensions

| | | |
|---|---|---|
| accuracy | granularity | fidelity |
| heterogeneity | precision | scalability |

- does a more complex model help understanding of complex systems?
- nothing new here? depends on
  - ratio of reactive : deliberative : generative behaviour
  - adaptation: hard-coded vs. data-driven behaviour
- policy is late-binding for simulation
- simulation is test environment for policy

## Why agent-based models?

- abstraction (equational) vs. synthesis (agent-based)
- ...or top-down vs. bottom-up
- approaches emphasize different dimensions

| | | |
|---|---|---|
| accuracy | granularity | fidelity |
| heterogeneity | precision | scalability |

- does a more complex model help understanding of complex systems?
- nothing new here? depends on
  - ratio of reactive : deliberative : generative behaviour
  - adaptation: hard-coded vs. data-driven behaviour
- policy is late-binding for simulation
- simulation is test environment for policy

## Why agent-based models?

- abstraction (equational) vs. synthesis (agent-based)
- ...or top-down vs. bottom-up
- approaches emphasize different dimensions

| | | |
|---|---|---|
| accuracy | granularity | fidelity |
| heterogeneity | precision | scalability |

- does a more complex model help understanding of complex systems?
- nothing new here? depends on
  - ratio of reactive : deliberative : generative behaviour
  - adaptation: hard-coded vs. data-driven behaviour
- policy is late-binding for simulation
- simulation is test environment for policy

## Why agent-based models?

- abstraction (equational) vs. synthesis (agent-based)
- ...or top-down vs. bottom-up
- approaches emphasize different dimensions

| | | |
|---|---|---|
| accuracy | granularity | fidelity |
| heterogeneity | precision | scalability |

- does a more complex model help understanding of complex systems?
- nothing new here? depends on
  - ratio of reactive : deliberative : generative behaviour
  - adaptation: hard-coded vs. data-driven behaviour
- policy is late-binding for simulation
- simulation is test environment for policy

## Why agent-based models?

- abstraction (equational) vs. synthesis (agent-based)
- ...or top-down vs. bottom-up
- approaches emphasize different dimensions

| | | |
|---|---|---|
| accuracy | granularity | fidelity |
| heterogeneity | precision | scalability |

- does a more complex model help understanding of complex systems?
- nothing new here? depends on
  - ratio of reactive : deliberative : generative behaviour
  - adaptation: hard-coded vs. data-driven behaviour
- policy is late-binding for simulation
- simulation is test environment for policy

## Why agent-based models?

- abstraction (equational) vs. synthesis (agent-based)
- ...or top-down vs. bottom-up
- approaches emphasize different dimensions

| | | |
|---|---|---|
| accuracy | granularity | fidelity |
| heterogeneity | precision | scalability |

- does a more complex model help understanding of complex systems?
- nothing new here? depends on
  - ratio of reactive : deliberative : generative behaviour
  - adaptation: hard-coded vs. data-driven behaviour
- policy is late-binding for simulation
- simulation is test environment for policy

## Why agent-based models?

- abstraction (equational) vs. synthesis (agent-based)
- ...or top-down vs. bottom-up
- approaches emphasize different dimensions

| | | |
|---|---|---|
| accuracy | granularity | fidelity |
| heterogeneity | precision | scalability |

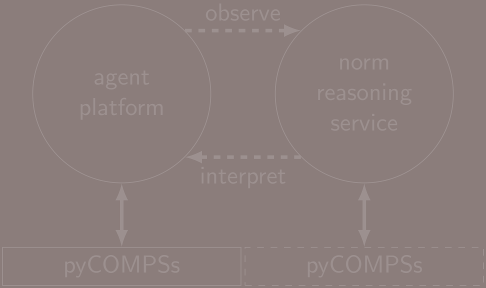
- does a more complex model help understanding of complex systems?
- nothing new here? depends on
  - ratio of reactive : deliberative : generative behaviour
  - adaptation: hard-coded vs. data-driven behaviour
- policy is late-binding for simulation
- simulation is test environment for policy

## Technology jigsaw

- Agent-based simulation constraints:

$$\begin{aligned}\text{sample size} &= \text{memory}\\ \text{serialization} &= \text{time}\\ \text{parameter range} + \text{dimensions} &= \text{time to sweep}\\ \text{simple (individual) models} &= \text{fidelity?}\end{aligned}$$

- Map to HPC? Overheads of many small tasks
- HPC opportunity: fidelity++ ⇒ better fit + all above

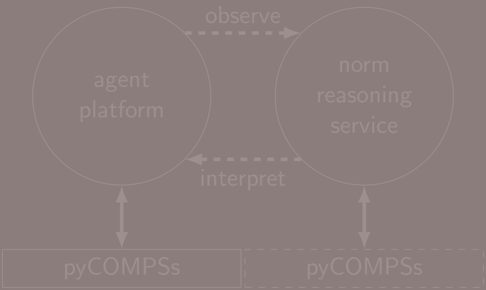


## Technology jigsaw

- Agent-based simulation constraints:

| | | |
|---|---|---|
| sample size | = | memory |
| serialization | = | time |
| parameter range + dimensions | = | time to sweep |
| simple (individual) models | = | fidelity? |

- Map to HPC? Overheads of many small tasks
- HPC opportunity: fidelity++ ⇒ better fit + all above

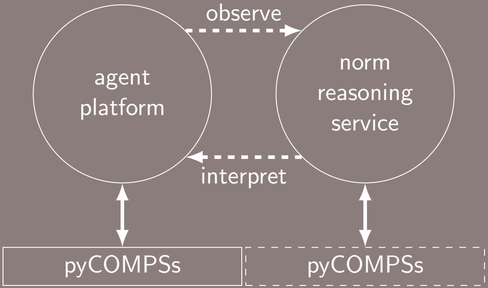


## Technology jigsaw

- Agent-based simulation constraints:

$$\begin{aligned} \text{sample size} &= \text{memory} \\ \text{serialization} &= \text{time} \\ \text{parameter range} + \text{dimensions} &= \text{time to sweep} \\ \text{simple (individual) models} &= \text{fidelity?} \end{aligned}$$

- Map to HPC? Overheads of many small tasks
- HPC opportunity: fidelity++ $\Rightarrow$ better fit + all above



## Contents

## Context

- capture (in)formal contextualized expectations of behaviour
  - what ought (not) to be true
  - what permissions (P) or prohibitions (F) hold
  - what obligations (O) hold
  - deontic logic (Wright 1951) of F, P, O
- ⇝ knowledge representation as norms
- ⇝ governance of agents in multiagent systems
- ⇝ governance of actors in socio-cognitive technical systems

## Context

- capture (in)formal contextualized expectations of behaviour
  - what ought (not) to be true
  - what permissions (P) or prohibitions (F) hold
  - what obligations (O) hold
  - deontic logic (Wright 1951) of F, P, O
- $\rightsquigarrow$ knowledge representation as norms
- $\rightsquigarrow$ governance of agents in multiagent systems
- $\rightsquigarrow$ governance of actors in socio-cognitive technical systems

## Context

- capture (in)formal contextualized expectations of behaviour
  - what ought (not) to be true
  - what permissions (P) or prohibitions (F) hold
  - what obligations (O) hold
  - deontic logic (Wright 1951) of F, P, O
- $\rightsquigarrow$ knowledge representation as norms
- $\rightsquigarrow$ governance of agents in multiagent systems
- $\rightsquigarrow$ governance of actors in socio-cognitive technical systems

## Context

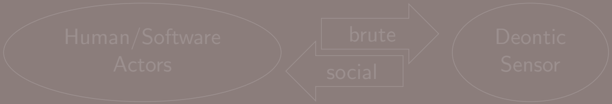
- capture (in)formal contextualized expectations of behaviour
  - what ought (not) to be true
  - what permissions (P) or prohibitions (F) hold
  - what obligations (O) hold
  - deontic logic (Wright 1951) of F, P, O
- $\rightsquigarrow$ knowledge representation as norms
- $\rightsquigarrow$ governance of agents in multiagent systems
- $\rightsquigarrow$ governance of actors in socio-cognitive technical systems

## Conceptual overview

- inspiration: { economics (North 1991) / social sciences (Harré et al. 1972) / social policy (Ostrom 1990) }
  - norm = constraint on action in a context
  - norm = part of policy or regulation or requirement
  - institution = set of norms
  - institution = policy or regulations or requirements
  - associates action with (institutional) consequences
- constitutive norms (Searle 1995):

  brute facts $\rightsquigarrow$ social facts
- counts-as (Jones et al. 1996):

  real-world event $\rightsquigarrow$ institutional event

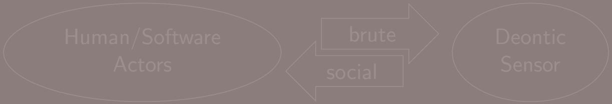


## Conceptual overview

- inspiration:
  - economics (North 1991)
  - social sciences (Harré et al. 1972)
  - social policy (Ostrom 1990)
  - norm = constraint on action in a context
  - norm = part of policy or regulation or requirement
  - institution = set of norms
  - institution = policy or regulations or requirements
  - associates action with (institutional) consequences
- constitutive norms (Searle 1995):

  brute facts $\rightsquigarrow$ social facts
- counts-as (Jones et al. 1996):

  real-world event $\rightsquigarrow$ institutional event



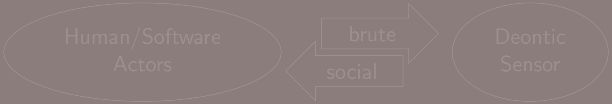
## Conceptual overview

- inspiration: { economics (North 1991); social sciences (Harré et al. 1972); social policy (Ostrom 1990) }
  - norm = constraint on action in a context
  - norm = part of policy or regulation or requirement
  - institution = set of norms
  - institution = policy or regulations or requirements
  - associates action with (institutional) consequences
- constitutive norms (Searle 1995):

$$\text{brute facts} \rightsquigarrow \text{social facts}$$

- counts-as (Jones et al. 1996):

$$\text{real-world event} \rightsquigarrow \text{institutional event}$$



## Conceptual overview

- inspiration: { economics (North 1991); social sciences (Harré et al. 1972); social policy (Ostrom 1990) }
    - norm = constraint on action in a context
    - norm = part of policy or regulation or requirement
    - institution = set of norms
    - institution = policy or regulations or requirements
    - associates action with (institutional) consequences
- constitutive norms (Searle 1995):

  brute facts $\rightsquigarrow$ social facts
- counts-as (Jones et al. 1996):

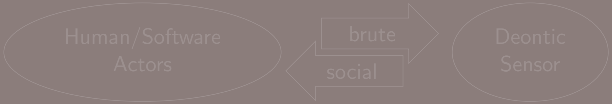
  real-world event $\rightsquigarrow$ institutional event

## Conceptual overview

- inspiration: { economics (North 1991); social sciences (Harré et al. 1972); social policy (Ostrom 1990) }
  - norm = constraint on action in a context
  - norm = part of policy or regulation or requirement
  - institution = set of norms
  - institution = policy or regulations or requirements
  - associates action with (institutional) consequences
- constitutive norms (Searle 1995):

  brute facts ⇝ social facts
- counts-as (Jones et al. 1996):

  real-world event ⇝ institutional event

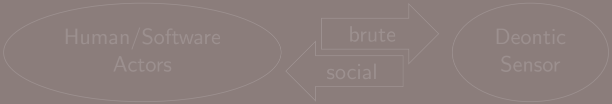


## Conceptual overview

- inspiration:
  - economics (North 1991)
  - social sciences (Harré et al. 1972)
  - social policy (Ostrom 1990)
  - norm = constraint on action in a context
  - norm = part of policy or regulation or requirement
  - institution = set of norms
  - institution = policy or regulations or requirements
  - associates action with (institutional) consequences
- constitutive norms (Searle 1995):

  brute facts $\rightsquigarrow$ social facts
- counts-as (Jones et al. 1996):

  real-world event $\rightsquigarrow$ institutional event

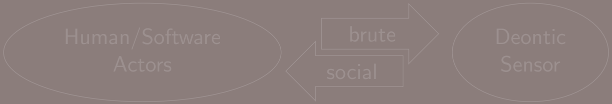


## Conceptual overview

- inspiration:
  - economics (North 1991)
  - social sciences (Harré et al. 1972)
  - social policy (Ostrom 1990)
  - norm = constraint on action in a context
  - norm = part of policy or regulation or requirement
  - institution = set of norms
  - institution = policy or regulations or requirements
  - associates action with (institutional) consequences
- constitutive norms (Searle 1995):

  brute facts $\rightsquigarrow$ social facts
- counts-as (Jones et al. 1996):

  real-world event $\rightsquigarrow$ institutional event

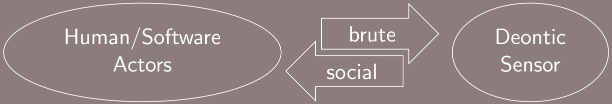


## Conceptual overview

- inspiration: { economics (North 1991); social sciences (Harré et al. 1972); social policy (Ostrom 1990) }
    - norm = constraint on action in a context
    - norm = part of policy or regulation or requirement
    - institution = set of norms
    - institution = policy or regulations or requirements
    - associates action with (institutional) consequences
- constitutive norms (Searle 1995):

  brute facts $\rightsquigarrow$ social facts
- counts-as (Jones et al. 1996):

  real-world event $\rightsquigarrow$ institutional event



## Actions change the (institutional) world

- counts-as: $\mathcal{G}: \text{external action} \xrightarrow[\text{if(conditions)}]{\text{generates}} \text{institutional action}$

- institutional facts represented by fluents
  1. fluent $\Rightarrow$ true if present, false otherwise

     $$\mathcal{C}^{\uparrow}: \text{action} \xrightarrow[\text{if(conditions)}]{\text{initiates}} \text{fluent}$$

  2. inertial fluent:

     $$\mathcal{C}^{\downarrow}: \text{action} \xrightarrow[\text{if(conditions)}]{\text{terminates}} \text{fluent}$$

     good for facts true for a period with start and finish actions

  3. non-inertial fluent: $\mathcal{C}^{N}: \xrightarrow[\text{if(conditions)}]{} \text{fluent}$

     good for facts expressed as combination of inertial + non-inertial fluents

## Actions change the (institutional) world

- counts-as: $\mathcal{G}$ : external action $\xrightarrow[\text{if(conditions)}]{\text{generates}}$ institutional action
- institutional facts represented by fluents
  1. fluent ⇒ true if present, false otherwise

     $\mathcal{C}^{\uparrow}$: action $\xrightarrow[\text{if(conditions)}]{\text{initiates}}$ fluent
  2. inertial fluent:

     $\mathcal{C}^{\downarrow}$: action $\xrightarrow[\text{if(conditions)}]{\text{terminates}}$ ~~fluent~~

     good for facts true for a period with start and finish actions
  3. non-inertial fluent: $\mathcal{C}^{\mathcal{N}}$ : $\xrightarrow[\text{if(conditions)}]{}$ fluent

     good for facts expressed as combination of inertial + non-inertial fluents

## Actions change the (institutional) world

- counts-as: $\mathcal{G}$ : external action $\xrightarrow[\text{if(conditions)}]{\text{generates}}$ institutional action
- institutional facts represented by fluents
  1. fluent ⇒ true if present, false otherwise
  2. inertial fluent:

     $\mathcal{C}^{\uparrow}$: action $\xrightarrow[\text{if(conditions)}]{\text{initiates}}$ fluent

     $\mathcal{C}^{\downarrow}$: action $\xrightarrow[\text{if(conditions)}]{\text{terminates}}$ ~~fluent~~

     good for facts true for a period with start and finish actions
  3. non-inertial fluent: $\mathcal{C}^{\mathcal{N}}$ : $\xrightarrow[\text{if(conditions)}]{}$ fluent

     good for facts expressed as combination of inertial + non-inertial fluents

## Actions change the (institutional) world

- counts-as: $\mathcal{G}$ : external action $\xrightarrow[\text{if(conditions)}]{\text{generates}}$ institutional action
- institutional facts represented by fluents
  1. fluent $\Rightarrow$ true if present, false otherwise

     $\mathcal{C}^{\uparrow}$: action $\xrightarrow[\text{if(conditions)}]{\text{initiates}}$ fluent
  2. inertial fluent:

     $\mathcal{C}^{\downarrow}$: action $\xrightarrow[\text{if(conditions)}]{\text{terminates}}$ ~~fluent~~

     good for facts true for a period with start and finish actions
  3. non-inertial fluent: $\mathcal{C}^{\mathcal{N}}$ : $\xrightarrow[\text{if(conditions)}]{}$ fluent

     good for facts expressed as combination of inertial + non-inertial fluents

## Actions change the (institutional) world

- counts-as: $\mathcal{G}$ : external action $\xrightarrow[\text{if(conditions)}]{\text{generates}}$ institutional action
- institutional facts represented by fluents
  1. fluent $\Rightarrow$ true if present, false otherwise

     $\mathcal{C}^{\uparrow}$: action $\xrightarrow[\text{if(conditions)}]{\text{initiates}}$ fluent
  2. inertial fluent:

     $\mathcal{C}^{\downarrow}$: action $\xrightarrow[\text{if(conditions)}]{\text{terminates}}$ ~~fluent~~

     good for facts true for a period with start and finish actions
  3. non-inertial fluent: $\mathcal{C}^{\mathcal{N}}$ : $\xrightarrow[\text{if(conditions)}]{}$ fluent

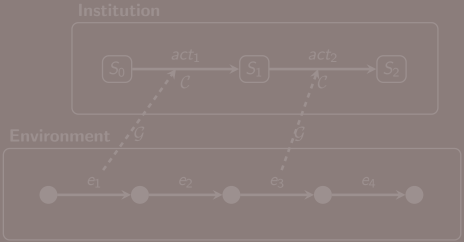
     good for facts expressed as combination of inertial + non-inertial fluents

## Making it work

- mathematical model:
  sets + relations $(\mathcal{G}, \mathcal{C}) \rightsquigarrow$ labelled transition system

$$\Delta \xrightarrow{e_1} S_1 \xrightarrow{e_2} S_2 \xrightarrow{e_3} \dots$$

- conceptual model:



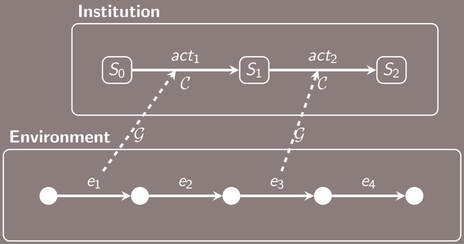

- generalizes to multiple, connected institutions

## Making it work

- mathematical model:
  sets + relations $(\mathcal{G}, \mathcal{C}) \rightsquigarrow$ labelled transition system

$$\Delta \xrightarrow{e_1} S_1 \xrightarrow{e_2} S_2 \xrightarrow{e_3} \dots$$

- conceptual model:



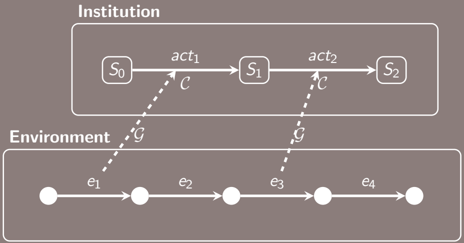

- generalizes to multiple, connected institutions

## Making it work

- mathematical model:
  sets + relations $(\mathcal{G}, \mathcal{C}) \rightsquigarrow$ labelled transition system

$$\Delta \xrightarrow{e_1} S_1 \xrightarrow{e_2} S_2 \xrightarrow{e_3} \ldots$$

- conceptual model:




- generalizes to multiple, connected institutions

## Implementation

- computational model:

  InstAL $\xrightarrow{\text{translator}}$ AnsProlog $\xrightarrow{\text{clingo}}$ answer sets $\xrightarrow{\text{visualisers}}$

- python front-end
- compiler: InstAL to Answer Set Programming
- python API to clingo (answer set solver, C++)
- answer sets delivered in JSON
- visualization tools generate images from traces

## Implementation

- computational model:

  InstAL $\xrightarrow{\text{translator}}$ AnsProlog $\xrightarrow{\text{clingo}}$ answer sets $\xrightarrow{\text{visualisers}}$

- python front-end
- compiler: InstAL to Answer Set Programming
- python API to clingo (answer set solver, C++)
- answer sets delivered in JSON
- visualization tools generate images from traces

## Implementation

- computational model:

InstAL --translator--> AnsProlog --clingo--> answer sets --visualisers-->

- python front-end
- compiler: InstAL to Answer Set Programming
- python API to clingo (answer set solver, C++)
- answer sets delivered in JSON
- visualization tools generate images from traces

## Implementation

- computational model:

InstAL $\xrightarrow{\text{translator}}$ AnsProlog $\xrightarrow{\text{clingo}}$ answer sets $\xrightarrow{\text{visualisers}}$

- python front-end
- compiler: InstAL to Answer Set Programming
- python API to clingo (answer set solver, C++)
- answer sets delivered in JSON
- visualization tools generate images from traces

## Implementation

- computational model:

InstAL $\xrightarrow{\text{translator}}$ AnsProlog $\xrightarrow{\text{clingo}}$ answer sets $\xrightarrow{\text{visualisers}}$

- python front-end
- compiler: InstAL to Answer Set Programming
- python API to clingo (answer set solver, C++)
- answer sets delivered in JSON
- visualization tools generate images from traces

## Implementation

- computational model:

  InstAL $\xrightarrow{\text{translator}}$ AnsProlog $\xrightarrow{\text{clingo}}$ answer sets $\xrightarrow{\text{visualisers}}$

- python front-end
- compiler: InstAL to Answer Set Programming
- python API to clingo (answer set solver, C++)
- answer sets delivered in JSON
- visualization tools generate images from traces

## Contents

## AI software siloes

- agent platform package =
  agent behaviour + environment + [institution(s)]
  see Aldewereld et al. (2016) for survey
- institutions absent or optional extra
- environment interface standard Behrens et al. (2011)
- buy the package: can't build platform from components

## AI software siloes

- agent platform package =
  agent behaviour + environment + [institution(s)]
  see Aldewereld et al. (2016) for survey
- institutions absent or optional extra
- environment interface standard Behrens et al. (2011)
- buy the package: can't build platform from components

## AI software siloes

- agent platform package =
  agent behaviour + environment + [institution(s)]
  see Aldewereld et al. (2016) for survey
- institutions absent or optional extra
- environment interface standard Behrens et al. (2011)
- buy the package: can't build platform from components

## AI software siloes

- agent platform package =
  agent behaviour + environment + [institution(s)]
  see Aldewereld et al. (2016) for survey
- institutions absent or optional extra
- environment interface standard Behrens et al. (2011)
- buy the package: can't build platform from components

## Software engineering approach

- refactoring: make norm reasoning a separate component
- design pattern: blueprint for deontic sensors
- resource-oriented architecture: deploy as RESTful services
- $\leadsto$ decoupling
- $\leadsto$ institution re-use
- $\leadsto$ institution certification
- access normative reasoning across + outside AI

## Software engineering approach

- refactoring: make norm reasoning a separate component
- design pattern: blueprint for deontic sensors
- resource-oriented architecture: deploy as RESTful services
- $\rightsquigarrow$ decoupling
- $\rightsquigarrow$ institution re-use
- $\rightsquigarrow$ institution certification
- access normative reasoning across + outside AI

## Software engineering approach

- refactoring: make norm reasoning a separate component
- design pattern: blueprint for deontic sensors
- resource-oriented architecture: deploy as RESTful services
- $\rightsquigarrow$ decoupling
- $\rightsquigarrow$ institution re-use
- $\rightsquigarrow$ institution certification
- access normative reasoning across + outside AI

## Software engineering approach

- refactoring: make norm reasoning a separate component
- design pattern: blueprint for deontic sensors
- resource-oriented architecture: deploy as RESTful services
- $\rightsquigarrow$ decoupling
- $\rightsquigarrow$ institution re-use
- $\rightsquigarrow$ institution certification
- access normative reasoning across + outside AI

## Software engineering approach

- refactoring: make norm reasoning a separate component
- design pattern: blueprint for deontic sensors
- resource-oriented architecture: deploy as RESTful services
- $\leadsto$ decoupling
- $\leadsto$ institution re-use
- $\leadsto$ institution certification
- access normative reasoning across + outside AI

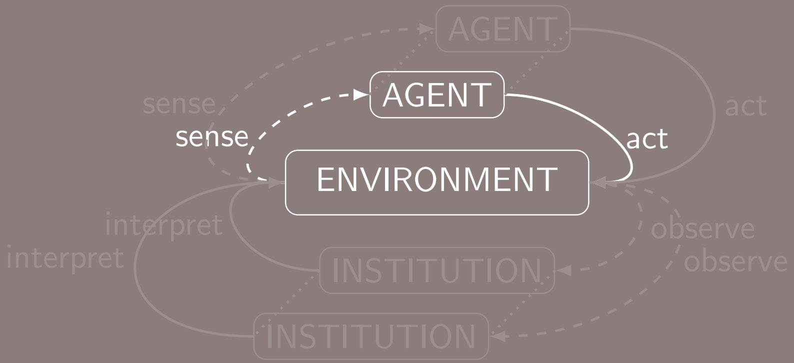
## Software engineering approach

- refactoring: make norm reasoning a separate component
- design pattern: blueprint for deontic sensors
- resource-oriented architecture: deploy as RESTful services
- $\rightsquigarrow$ decoupling
- $\rightsquigarrow$ institution re-use
- $\rightsquigarrow$ institution certification
- access normative reasoning across + outside AI

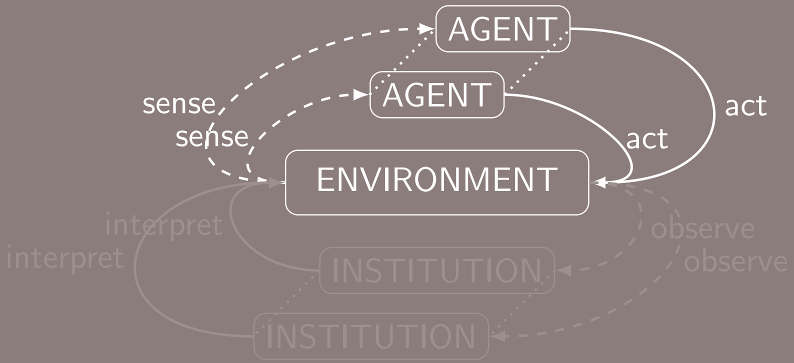
## Software engineering approach

- refactoring: make norm reasoning a separate component
- design pattern: blueprint for deontic sensors
- resource-oriented architecture: deploy as RESTful services
- $\rightsquigarrow$ decoupling
- $\rightsquigarrow$ institution re-use
- $\rightsquigarrow$ institution certification
- access normative reasoning across + outside AI

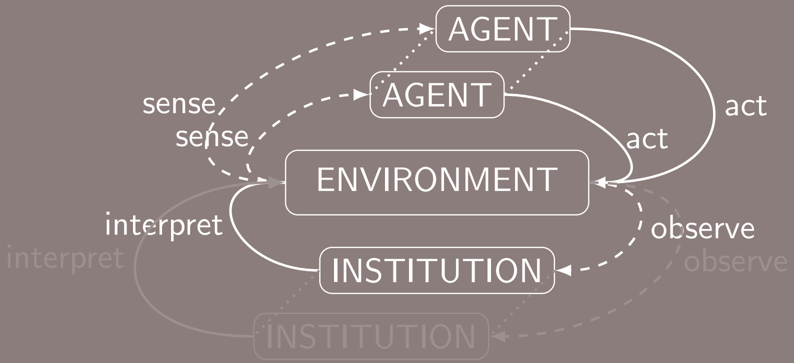
# Sense-Act + Observe-Interpret



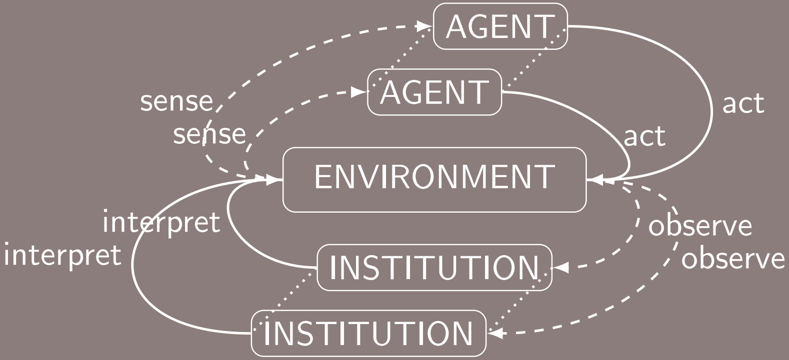
# Sense-Act + Observe-Interpret



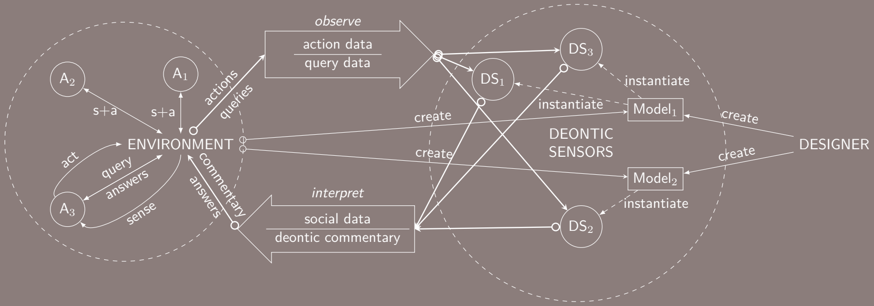
## Sense-Act + Observe-Interpret



## Sense-Act + Observe-Interpret

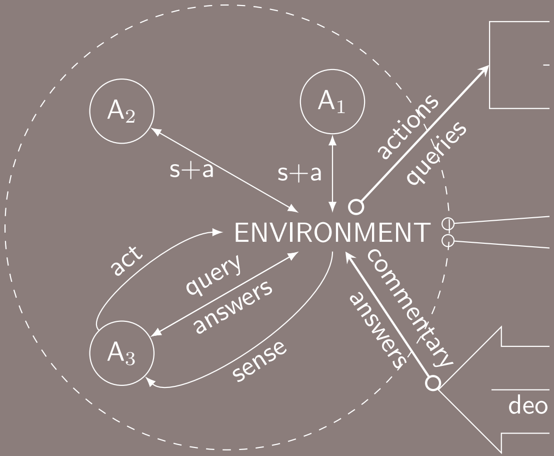


# Deontic Sensors Architecture

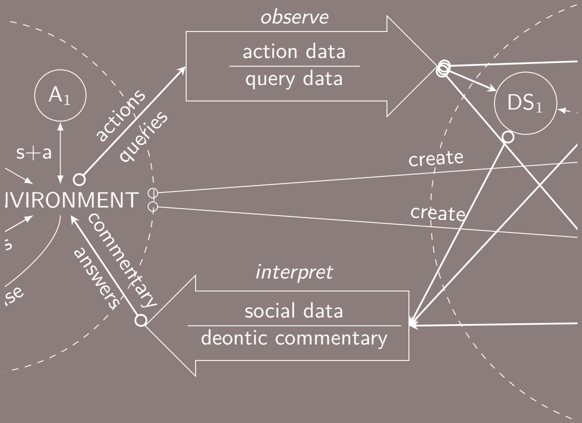



resource-oriented architecture (ROA) pattern for normative reasoning services (Padget et al. 2018)

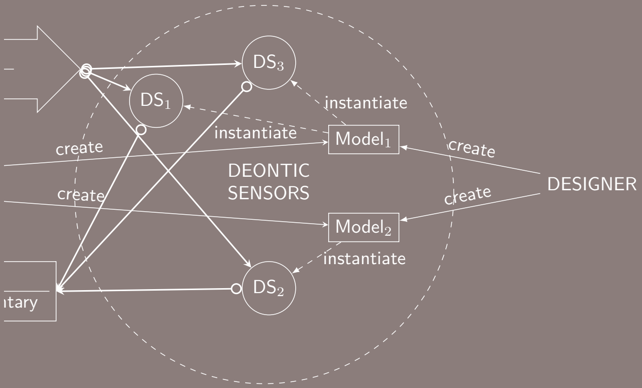
# Architecture: environment and agents



# Architecture: observe-interpret



# Architecture: deontic sensors



## ROA endpoints

1. → `POST /model/`

   *Create model from specification*

   ← `/model/`*X*

2. → `POST /model/`*X*`/instance/`

   *Create instance of model X with* `POST` *data*

   ← `/model/`*X*`/instance/`*Y*

3. → `POST /model/`*X*`/instance/`*Y*`/query/`

   *Create a query of instance Y with* `POST` *data*

   ← `/model/`*X*`/instantiate/`*Y*`/query/`*Z*

4. → `GET /model/`*X*`/instance/`*Y*`/query/`*Z*`/output`

   *Read result of query*

   ← result of query *Z* in a protocol-defined format

## ROA endpoints

1. → `POST /model/`

   *Create model from specification*

   ← `/model/`*X*

2. → `POST /model/`*X*`/instance/`

   *Create instance of model X with* `POST` *data*

   ← `/model/`*X*`/instance/`*Y*

3. → `POST /model/`*X*`/instance/`*Y*`/query/`

   *Create a query of instance Y with* `POST` *data*

   ← `/model/`*X*`/instantiate/`*Y*`/query/`*Z*

4. → `GET /model/`*X*`/instance/`*Y*`/query/`*Z*`/output`

   *Read result of query*

   ← result of query *Z* in a protocol-defined format

## ROA endpoints

1. → `POST /model/`

   *Create model from specification*

   ← `/model/`*X*

2. → `POST /model/`*X*`/instance/`

   *Create instance of model X with* `POST` *data*

   ← `/model/`*X*`/instance/`*Y*

3. → `POST /model/`*X*`/instance/`*Y*`/query/`

   *Create a query of instance Y with* `POST` *data*

   ← `/model/`*X*`/instantiate/`*Y*`/query/`*Z*

4. → `GET /model/`*X*`/instance/`*Y*`/query/`*Z*`/output`

   *Read result of query*

   ← result of query *Z* in a protocol-defined format

## ROA endpoints

1. → `POST /model/`

   *Create model from specification*

   ← `/model/`*X*

2. → `POST /model/`*X*`/instance/`

   *Create instance of model X with* `POST` *data*

   ← `/model/`*X*`/instance/`*Y*

3. → `POST /model/`*X*`/instance/`*Y*`/query/`

   *Create a query of instance Y with* `POST` *data*

   ← `/model/`*X*`/instantiate/`*Y*`/query/`*Z*

4. → `GET /model/`*X*`/instance/`*Y*`/query/`*Z*`/output`

   *Read result of query*

   ← result of query *Z* in a protocol-defined format

## Instantiating the pattern

- agent platform: Jason (Bordini et al. 2007)
  - Belief-Desire-Intention (BDI) agent architecture
  - means-end reasoning
  - open-minded commitment
- normative reasoning: InstAL (Padget et al. 2016a)
  - InstAL: **Inst**itutional **A**ction **L**anguage
  - builds model in Answer Set Prolog
  - symbolic model checking
    - single event $\rightarrow$ new model state: +/- FPO +/- domain facts
    - multiple events $\rightarrow$ alternative model states
- InstAL as a service:
  - python Flask for RESTful API
  - celery + rabbitMQ for server task creation/communications
  - python InstAL compiler
  - clingo for grounding + solving
  - python client: JSON payload both ways

## Instantiating the pattern

- agent platform: Jason (Bordini et al. 2007)
  - Belief-Desire-Intention (BDI) agent architecture
  - means-end reasoning
  - open-minded commitment
- normative reasoning: InstAL (Padget et al. 2016a)
  - InstAL: **Inst**itutional **A**ction **L**anguage
  - builds model in Answer Set Prolog
  - symbolic model checking
    - single event $\rightarrow$ new model state: +/- FPO +/- domain facts
    - multiple events $\rightarrow$ alternative model states
- InstAL as a service:
  - python Flask for RESTful API
  - celery + rabbitMQ for server task creation/communications
  - python InstAL compiler
  - clingo for grounding + solving
  - python client: JSON payload both ways

## Instantiating the pattern

- agent platform: Jason (Bordini et al. 2007)
  - Belief-Desire-Intention (BDI) agent architecture
  - means-end reasoning
  - open-minded commitment
- normative reasoning: InstAL (Padget et al. 2016a)
  - InstAL: **Inst**itutional **A**ction **L**anguage
  - builds model in Answer Set Prolog
  - symbolic model checking
    - single event → new model state: +/- FPO +/- domain facts
    - multiple events → alternative model states
- InstAL as a service:
  - python Flask for RESTful API
  - celery + rabbitMQ for server task creation/communications
  - python InstAL compiler
  - clingo for grounding + solving
  - python client: JSON payload both ways

## Instantiating the pattern

- agent platform: Jason (Bordini et al. 2007)
  - Belief-Desire-Intention (BDI) agent architecture
  - means-end reasoning
  - open-minded commitment
- normative reasoning: InstAL (Padget et al. 2016a)
  - InstAL: **Inst**itutional **A**ction **L**anguage
  - builds model in Answer Set Prolog
  - symbolic model checking
    - single event → new model state: +/- FPO +/- domain facts
    - multiple events → alternative model states
- InstAL as a service:
  - python Flask for RESTful API
  - celery + rabbitMQ for server task creation/communications
  - python InstAL compiler
  - clingo for grounding + solving
  - python client: JSON payload both ways

## Instantiating the pattern

- agent platform: Jason (Bordini et al. 2007)
  - Belief-Desire-Intention (BDI) agent architecture
  - means-end reasoning
  - open-minded commitment
- normative reasoning: InstAL (Padget et al. 2016a)
  - InstAL: **Inst**itutional **A**ction **L**anguage
  - builds model in Answer Set Prolog
  - symbolic model checking
    - single event → new model state: +/- FPO +/- domain facts
    - multiple events → alternative model states
- InstAL as a service:
  - python Flask for RESTful API
  - celery + rabbitMQ for server task creation/communications
  - python InstAL compiler
  - clingo for grounding + solving
  - python client: JSON payload both ways

## Instantiating the pattern

- agent platform: Jason (Bordini et al. 2007)
  - Belief-Desire-Intention (BDI) agent architecture
  - means-end reasoning
  - open-minded commitment
- normative reasoning: InstAL (Padget et al. 2016a)
  - InstAL: **Inst**itutional **A**ction **L**anguage
  - builds model in Answer Set Prolog
  - symbolic model checking
    - single event → new model state: +/- FPO +/- domain facts
    - multiple events → alternative model states
- InstAL as a service:
  - python Flask for RESTful API
  - celery + rabbitMQ for server task creation/communications
  - python InstAL compiler
  - clingo for grounding + solving
  - python client: JSON payload both ways

## Instantiating the pattern

- agent platform: Jason (Bordini et al. 2007)
  - Belief-Desire-Intention (BDI) agent architecture
  - means-end reasoning
  - open-minded commitment
- normative reasoning: InstAL (Padget et al. 2016a)
  - InstAL: **Inst**itutional **A**ction **L**anguage
  - builds model in Answer Set Prolog
  - symbolic model checking
    - single event → new model state: +/- FPO +/- domain facts
    - multiple events → alternative model states
- InstAL as a service:
  - python Flask for RESTful API
  - celery + rabbitMQ for server task creation/communications
  - python InstAL compiler
  - clingo for grounding + solving
  - python client: JSON payload both ways

## Instantiating the pattern

- agent platform: Jason (Bordini et al. 2007)
  - Belief-Desire-Intention (BDI) agent architecture
  - means-end reasoning
  - open-minded commitment
- normative reasoning: InstAL (Padget et al. 2016a)
  - InstAL: **Inst**itutional **A**ction **L**anguage
  - builds model in Answer Set Prolog
  - symbolic model checking
    - single event → new model state: +/- FPO +/- domain facts
    - multiple events → alternative model states
- InstAL as a service:
  - python Flask for RESTful API
  - celery + rabbitMQ for server task creation/communications
  - python InstAL compiler
  - clingo for grounding + solving
  - python client: JSON payload both ways

## Instantiating for HPC

1. replace Celery with pyCOMPSs
2. replace Flask (RESTful) API with conventional API
3. extend agent platform Controller to
   - send actions to InstAL
   - receive interpretations from InstAL
   - publish interpretation for agents to perceive
4. extend agent reasoning to account for normative percepts

## Instantiating for HPC

1. replace Celery with pyCOMPSs
2. replace Flask (RESTful) API with conventional API
3. extend agent platform Controller to
   - send actions to InstAL
   - receive interpretations from InstAL
   - publish interpretation for agents to perceive
4. extend agent reasoning to account for normative percepts

## Instantiating for HPC

1. replace Celery with pyCOMPSs
2. replace Flask (RESTful) API with conventional API
3. extend agent platform Controller to
   - send actions to InstAL
   - receive interpretations from InstAL
   - publish interpretation for agents to perceive
4. extend agent reasoning to account for normative percepts

## Instantiating for HPC

1. replace Celery with pyCOMPSs
2. replace Flask (RESTful) API with conventional API
3. extend agent platform Controller to
   - send actions to InstAL
   - receive interpretations from InstAL
   - publish interpretation for agents to perceive
4. extend agent reasoning to account for normative percepts

## Instantiating for HPC

1. replace Celery with pyCOMPSs
2. replace Flask (RESTful) API with conventional API
3. extend agent platform Controller to
   - send actions to InstAL
   - receive interpretations from InstAL
   - publish interpretation for agents to perceive
4. extend agent reasoning to account for normative percepts

## Instantiating for HPC

1. replace Celery with pyCOMPSs
2. replace Flask (RESTful) API with conventional API
3. extend agent platform Controller to
   - send actions to InstAL
   - receive interpretations from InstAL
   - publish interpretation for agents to perceive
4. extend agent reasoning to account for normative percepts

## Instantiating for HPC

1. replace Celery with pyCOMPSs
2. replace Flask (RESTful) API with conventional API
3. extend agent platform Controller to
   - send actions to InstAL
   - receive interpretations from InstAL
   - publish interpretation for agents to perceive
4. extend agent reasoning to account for normative percepts

# Contents

## InstAL declarations

Different types of declarations:

```
type Industry;
exogenous event discharge(WWTP,Mass);
violation event illegalDischarge(WWTP,Mass);
inst event iDischarge(WWTP,Mass);
fluent highMercury(Mass);
obligation fluent obl(
   iInform(Industry,WWTP,Mass),          % event
   iRelease(Industry,WWTP,Mass),         % deadline
   failureToInform(Industry,WWTP,Mass)); % violation
```

# InstAL rules

Generates, initiates and terminates rules:

```
discharge(WWTP,Mass) generates iDischarge(WWTP,Mass)
   if treated(WWTP,Mass,Treatment);
discharge(WWTP,Mass) generates illegalDischarge(WWTP,Mass)
   if not treated(WWTP,Mass,Treatment);
discharge(WWTP,Mass) generates illegalDischarge(WWTP,Mass)
   if highMercury(Mass);
illegalDischarge(WWTP,Mass) initiates illegalBecause(untreated,
    WWTP,Mass)
   if not treated(WWTP,Mass,Treatment);
illegalDischarge(WWTP,Mass) initiates illegalBecause(
    high_mercury,WWTP,Mass)
   if highMercury(Mass);
iDischarge(WWTP,Mass) terminates treated(WWTP,Mass,Treatment)
   if treated(WWTP,Mass,Treatment), not highMercury(Mass);
iPerform(WWTP,Mass,Treatment) initiates treated(WWTP,Mass,
    Treatment)
   if treating(WWTP,Mass);
initially
highMercury(m2),
signedContract(wwtp1,i1),
obl(iInform(i1,wwtp1,M),iRelease(i1,wwtp1,M),failureToInform(i1,
    wwtp1,M))
```

## Sample run

- grounding specification:

```
Industry: i1 i2
Mass: m1 m2
Reason: untreated high_mercury
Treatment: tk
WWTP: wwtp1 wwtp2
```

- input trace:

```
observed(inform(i1,wwtp1,m2))
observed(release(i1,wwtp1,m2))
observed(receive(wwtp1,i1,m2))
observed(perform(wwtp1,m2,tk))
observed(discharge(wwtp1,m2))
observed(release(i2,wwtp2,m1))
observed(receive(wwtp2,i2,m1))
observed(discharge(wwtp2,m1))
```

## Sample run

- grounding specification:

```
Industry: i1 i2
Mass: m1 m2
Reason: untreated high_mercury
Treatment: tk
WWTP: wwtp1 wwtp2
```

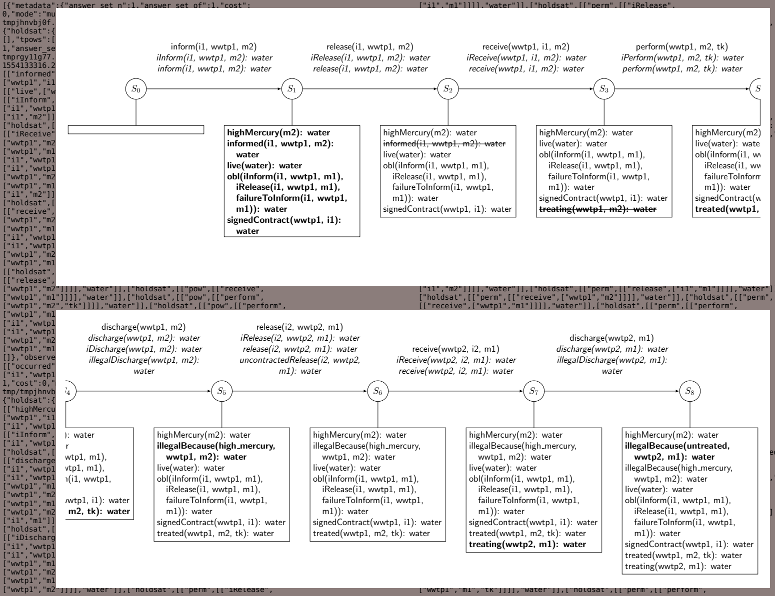
- input trace:

```
observed(inform(i1,wwtp1,m2))
observed(release(i1,wwtp1,m2))
observed(receive(wwtp1,i1,m2))
observed(perform(wwtp1,m2,tk))
observed(discharge(wwtp1,m2))
observed(release(i2,wwtp2,m1))
observed(receive(wwtp2,i2,m1))
observed(discharge(wwtp2,m1))
```

# Trace visualization

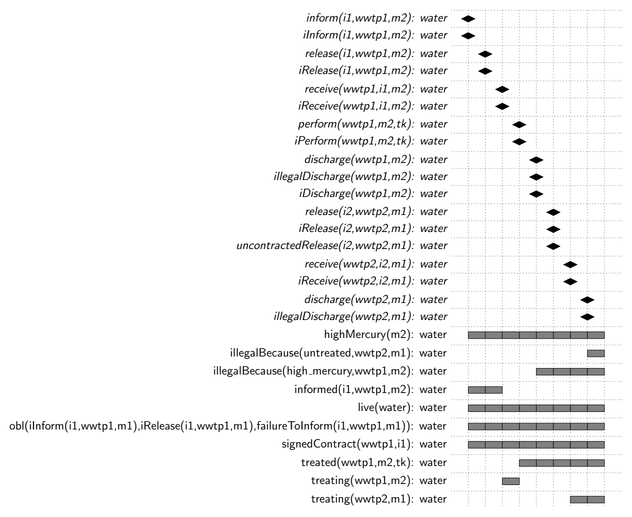

[{"metadata":{"answer_set_n":1,"answer_set_of":1,"cost":
0,"mode":"multi_shot","pid":7887,"source_files":["/tmp/tmprgy11g77.lp","/tmp/
tmpjhnvbj0f.lp"],"timestamp":1554133316.2317219,"version":"1.0.1"},"state":
{"holdsat":{"fluents":[],"gpows":[],"ipows":[],"obls":[],"perms":[],"pows":
[],"tpows":[]},"observed":[],"occurred":[]}},{"metadata":{"answer_set_n":
1,"answer_set_of":1,"cost":0,"mode":"multi_shot","pid":7887,"source_files":["/tmp/
tmprgy11g77.lp","/tmp/tmpjhnvbj0f.lp"],"timestamp":
1554133316.2317219,"version":"1.0.1"},"state":{"holdsat":{"fluents":[["holdsat",
[["informed",["i1","wwtp1","m2"]],"water"]],["holdsat",[["signedContract",
["wwtp1","i1"]],"water"]],["holdsat",[["highMercury",["m2"]],"water"]],["holdsat",
[["live",["water"]],"water"]]],"gpows":[],"ipows":[],"obls":[["holdsat",[["obl",
[["iInform",["i1","wwtp1","m1"]],["iRelease",["i1","m1"]],["failureToInform",
["i1","wwtp1","m1"]]]],"water"]]],"perms":[["holdsat",[["perm",[["iRelease",
["i1","m2"]]]],"water"]],["holdsat",[["perm",[["iRelease",["i1","m1"]]]],"water"]],
["holdsat",[["perm",[["iReceive",["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",
[["iReceive",["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["iPerform",
["wwtp1","m2","tk"]]]],"water"]],["holdsat",[["perm",[["iPerform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["perm",[["iInform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["iInform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["iDischarge",
["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["iDischarge",
["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["release",
["i1","m2"]]]],"water"]],["holdsat",[["perm",[["release",["i1","m1"]]]],"water"]],
["holdsat",[["perm",[["receive",["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",
[["receive",["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["perform",
["wwtp1","m2","tk"]]]],"water"]],["holdsat",[["perm",[["perform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["perm",[["inform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["inform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["discharge",
["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["discharge",
["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",["null"]],"water"]]],"pows":
[["holdsat",[["pow",[["release",["i1","m2"]]]],"water"]],["holdsat",[["pow",
[["release",["i1","m1"]]]],"water"]],["holdsat",[["pow",[["receive",
["wwtp1","m2"]]]],"water"]],["holdsat",[["pow",[["receive",
["wwtp1","m1"]]]],"water"]],["holdsat",[["pow",[["perform",
["wwtp1","m2","tk"]]]],"water"]],["holdsat",[["pow",[["perform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["pow",[["inform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["pow",[["inform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["pow",[["discharge",
["wwtp1","m2"]]]],"water"]],["holdsat",[["pow",[["discharge",
["wwtp1","m1"]]]],"water"]],["holdsat",[["pow",["null"]],"water"]]],"tpows":
[]},"observed":[["observed",[["inform",["i1","wwtp1","m2"]]]]],"occurred":
[["occurred",[["inform",["i1","wwtp1","m2"]],"water"]],["occurred",[["iInform",
["i1","wwtp1","m2"]],"water"]]]}},{"metadata":{"answer_set_n":1,"answer_set_of":
1,"cost":0,"mode":"multi_shot","pid":7887,"source_files":["/tmp/tmprgy11g77.lp","/
tmp/tmpjhnvbj0f.lp"],"timestamp":1554133316.2317219,"version":"1.0.1"},"state":
{"holdsat":{"fluents":[["holdsat",[["live",["water"]],"water"]],["holdsat",
[["highMercury",["m2"]],"water"]],["holdsat",[["signedContract",
["wwtp1","i1"]],"water"]],["holdsat",[["informed",
["i1","wwtp1","m2"]],"water"]]],"gpows":[],"ipows":[],"obls":[["holdsat",[["obl",
[["iInform",["i1","wwtp1","m1"]],["iRelease",["i1","m1"]],["failureToInform",
["i1","wwtp1","m1"]]]],"water"]]],"perms":[["holdsat",[["perm",["null"]],"water"]],
["holdsat",[["perm",[["discharge",["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",
[["discharge",["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["inform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["inform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["perform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["perm",[["perform",
["wwtp1","m2","tk"]]]],"water"]],["holdsat",[["perm",[["receive",
["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["receive",
["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["release",
["i1","m1"]]]],"water"]],["holdsat",[["perm",[["release",["i1","m2"]]]],"water"]],
["holdsat",[["perm",[["iDischarge",["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",
[["iDischarge",["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["iInform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["iInform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["iPerform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["perm",[["iPerform",
["wwtp1","m2","tk"]]]],"water"]],["holdsat",[["perm",[["iReceive",
["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["iReceive",
["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["iRelease",
["i1","m1"]]]],"water"]],["holdsat",[["perm",[["iRelease",
["i1","m2"]]]],"water"]]],"pows":[["holdsat",[["pow",["null"]],"water"]],
["holdsat",[["pow",[["discharge",["wwtp1","m1"]]]],"water"]],["holdsat",[["pow",
[["discharge",["wwtp1","m2"]]]],"water"]],["holdsat",[["pow",[["inform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["pow",[["inform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["pow",[["perform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["pow",[["perform",
["wwtp1","m2","tk"]]]],"water"]],["holdsat",[["pow",[["receive",
["wwtp1","m1"]]]],"water"]],["holdsat",[["pow",[["receive",
["wwtp1","m2"]]]],"water"]],["holdsat",[["pow",[["release",
["i1","m1"]]]],"water"]],["holdsat",[["pow",[["release",
["i1","m2"]]]],"water"]]],"tpows":[]},"observed":[["observed",[["release",
["i1","m2"]]]]],"occurred":[["occurred",[["release",["i1","m2"]],"water"]],
["occurred",[["iRelease",["i1","m2"]],"water"]],["occurred",[["failureToInform",
["i1","wwtp1","m2"]],"water"]]]}},{"metadata":{"answer_set_n":1,"answer_set_of":
1,"cost":0,"mode":"multi_shot","pid":7887,"source_files":["/tmp/tmprgy11g77.lp","/
tmp/tmpjhnvbj0f.lp"],"timestamp":1554133316.2317219,"version":"1.0.1"},"state":
{"holdsat":{"fluents":[["holdsat",[["treating",["wwtp1","m2"]],"water"]],
["holdsat",[["informed",["i1","wwtp1","m2"]],"water"]],["holdsat",
[["signedContract",["wwtp1","i1"]],"water"]],["holdsat",[["highMercury",
["m2"]],"water"]],["holdsat",[["live",["water"]],"water"]]],"gpows":[],"ipows":
[],"obls":[["holdsat",[["obl",[["iInform",["i1","wwtp1","m1"]],["iRelease",
["i1","m1"]],["failureToInform",["i1","wwtp1","m1"]]]],"water"]]],"perms":
[["holdsat",[["perm",[["iRelease",["i1","m2"]]]],"water"]],["holdsat",[["perm",
[["iRelease",["i1","m1"]]]],"water"]],["holdsat",[["perm",[["iReceive",
["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["iReceive",
["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["iPerform",
["wwtp1","m2","tk"]]]],"water"]],["holdsat",[["perm",[["iPerform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["perm",[["iInform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["iInform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["iDischarge",
["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["iDischarge",
["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["release",
["i1","m2"]]]],"water"]],["holdsat",[["perm",[["release",["i1","m1"]]]],"water"]],
["holdsat",[["perm",[["receive",["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",
[["receive",["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["perform",
["wwtp1","m2","tk"]]]],"water"]],["holdsat",[["perm",[["perform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["perm",[["inform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["inform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["discharge",
["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["discharge",
["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",["null"]],"water"]]],"pows":
[["holdsat",[["pow",[["release",["i1","m2"]]]],"water"]],["holdsat",[["pow",
[["release",["i1","m1"]]]],"water"]],["holdsat",[["pow",[["receive",
["wwtp1","m2"]]]],"water"]],["holdsat",[["pow",[["receive",
["wwtp1","m1"]]]],"water"]],["holdsat",[["pow",[["perform",
["wwtp1","m2","tk"]]]],"water"]],["holdsat",[["pow",[["perform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["pow",[["inform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["pow",[["inform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["pow",[["discharge",
["wwtp1","m2"]]]],"water"]],["holdsat",[["pow",[["discharge",
["wwtp1","m1"]]]],"water"]],["holdsat",[["pow",["null"]],"water"]]],"tpows":
[]},"observed":[["observed",[["receive",["wwtp1","m2"]]]]],"occurred":[["occurred",
[["receive",["wwtp1","m2"]],"water"]],["occurred",[["iReceive",
["wwtp1","m2"]],"water"]]]}},{"metadata":{"answer_set_n":1,"answer_set_of":
1,"cost":0,"mode":"multi_shot","pid":7887,"source_files":["/tmp/tmprgy11g77.lp","/
tmp/tmpjhnvbj0f.lp"],"timestamp":1554133316.2317219,"version":"1.0.1"},"state":
{"holdsat":{"fluents":[["holdsat",[["live",["water"]],"water"]],["holdsat",
[["highMercury",["m2"]],"water"]],["holdsat",[["signedContract",
["wwtp1","i1"]],"water"]],["holdsat",[["informed",["i1","wwtp1","m2"]],"water"]],
["holdsat",[["treated",["wwtp1","m2","tk"]],"water"]]],"gpows":[],"ipows":
[],"obls":[["holdsat",[["obl",[["iInform",["i1","wwtp1","m1"]],["iRelease",
["i1","m1"]],["failureToInform",["i1","wwtp1","m1"]]]],"water"]]],"perms":
[["holdsat",[["perm",["null"]],"water"]],["holdsat",[["perm",[["discharge",
["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["discharge",
["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["inform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["inform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["perform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["perm",[["perform",

# Trace visualization



# Trace visualization

[{"metadata":{"answer_set_n":1,"answer_set_of":1,"cost":
0,"mode":"multi_shot","pid":7887,"source_files":["/tmp/tmprgy11g77.lp","/tmp/
tmpjhnvbj0f.lp"],"timestamp":1554133316.2317219,"version":"1.0.1"},"state":
{"holdsat":{"fluents":[],"gpows":[],"ipows":[],"obls":[],"perms":[],"pows":
["i1","m1"]]]],"water"]],["holdsat",[["perm",[["iRelease",
["i1","m2"]]]],"water"]]],"pows":[["holdsat",[["pow",["null"]],"water"]],
["holdsat",[["pow",[["discharge",["wwtp1","m1"]]]],"water"]],["holdsat",[["pow",
[["discharge",["wwtp1","m2"]]]],"water"]],["holdsat",[["pow",[["inform",




["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["perm",[["iReceive",
["wwtp1","m2","tk"]]]],"water"]],["holdsat",[["perm",[["iReceive",
["wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["iReceive",
["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["iRelease",
["wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["inform",
["i1","wwtp1","m1"]]]],"water"]],["holdsat",[["perm",[["inform",
["i1","wwtp1","m2"]]]],"water"]],["holdsat",[["perm",[["perform",
["wwtp1","m1","tk"]]]],"water"]],["holdsat",[["perm",[["perform",

## Contents

## Where's the semantics?

- unfortunately, each model is "just another program"
- where "illegalDischarge" is just a string
- and the implementation is the programmer's interpretation
- need to connect real world to model (automatically)
- natural language → model?
- semantic representation of policy → model?
- W3C's Open Digital Rights Language (ODRL)
- Originally conceived for asset rights management: early 2000s
- ODRL 2.2 (2018) generalizes to policy... to some degree

## Where's the semantics?

- unfortunately, each model is "just another program"
- where "illegalDischarge" is just a string
- and the implementation is the programmer's interpretation
- need to connect real world to model (automatically)
- natural language → model?
- semantic representation of policy → model?
- W3C's Open Digital Rights Language (ODRL)
- Originally conceived for asset rights management: early 2000s
- ODRL 2.2 (2018) generalizes to policy... to some degree

## Where's the semantics?

- unfortunately, each model is "just another program"
- where "illegalDischarge" is just a string
- and the implementation is the programmer's interpretation
- need to connect real world to model (automatically)
- natural language $\rightarrow$ model?
- semantic representation of policy $\rightarrow$ model?
- W3C's Open Digital Rights Language (ODRL)
- Originally conceived for asset rights management: early 2000s
- ODRL 2.2 (2018) generalizes to policy... to some degree

## Where's the semantics?

- unfortunately, each model is "just another program"
- where "illegalDischarge" is just a string
- and the implementation is the programmer's interpretation
- need to connect real world to model (automatically)
- natural language → model?
- semantic representation of policy → model?
- W3C's Open Digital Rights Language (ODRL)
- Originally conceived for asset rights management: early 2000s
- ODRL 2.2 (2018) generalizes to policy... to some degree

## Where's the semantics?

- unfortunately, each model is "just another program"
- where "illegalDischarge" is just a string
- and the implementation is the programmer's interpretation
- need to connect real world to model (automatically)
- natural language $\rightarrow$ model?
- semantic representation of policy $\rightarrow$ model?
- W3C's Open Digital Rights Language (ODRL)
- Originally conceived for asset rights management: early 2000s
- ODRL 2.2 (2018) generalizes to policy... to some degree

## Where's the semantics?

- unfortunately, each model is "just another program"
- where "illegalDischarge" is just a string
- and the implementation is the programmer's interpretation
- need to connect real world to model (automatically)
- natural language $\rightarrow$ model?
- semantic representation of policy $\rightarrow$ model?
- W3C's Open Digital Rights Language (ODRL)
- Originally conceived for asset rights management: early 2000s
- ODRL 2.2 (2018) generalizes to policy... to some degree

## Where's the semantics?

- unfortunately, each model is "just another program"
- where "illegalDischarge" is just a string
- and the implementation is the programmer's interpretation
- need to connect real world to model (automatically)
- natural language $\rightarrow$ model?
- semantic representation of policy $\rightarrow$ model?
- W3C's Open Digital Rights Language (ODRL)
- Originally conceived for asset rights management: early 2000s
- ODRL 2.2 (2018) generalizes to policy... to some degree

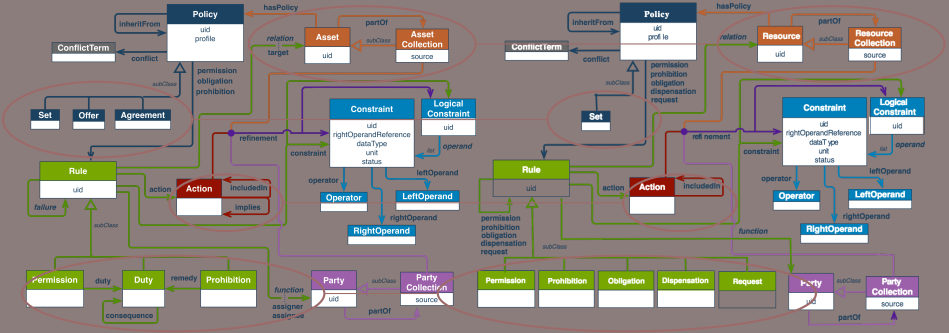
## Where's the semantics?

- unfortunately, each model is "just another program"
- where "illegalDischarge" is just a string
- and the implementation is the programmer's interpretation
- need to connect real world to model (automatically)
- natural language $\rightarrow$ model?
- semantic representation of policy $\rightarrow$ model?
- W3C's Open Digital Rights Language (ODRL)
- Originally conceived for asset rights management: early 2000s
- ODRL 2.2 (2018) generalizes to policy... to some degree

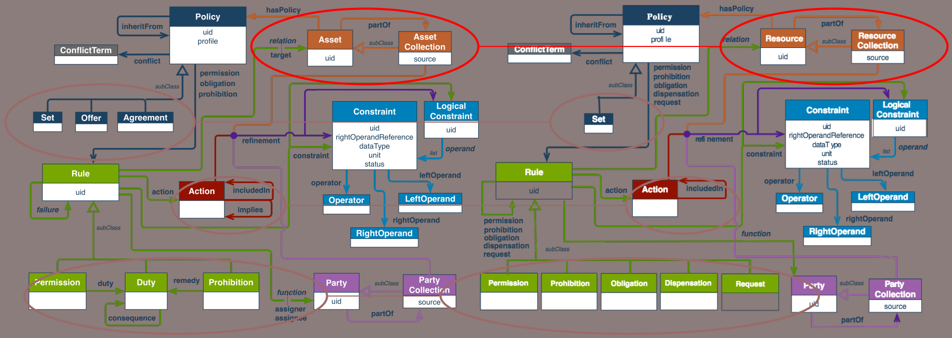
## Where's the semantics?

- unfortunately, each model is "just another program"
- where "illegalDischarge" is just a string
- and the implementation is the programmer's interpretation
- need to connect real world to model (automatically)
- natural language $\rightarrow$ model?
- semantic representation of policy $\rightarrow$ model?
- W3C's Open Digital Rights Language (ODRL)
- Originally conceived for asset rights management: early 2000s
- ODRL 2.2 (2018) generalizes to policy... to some degree

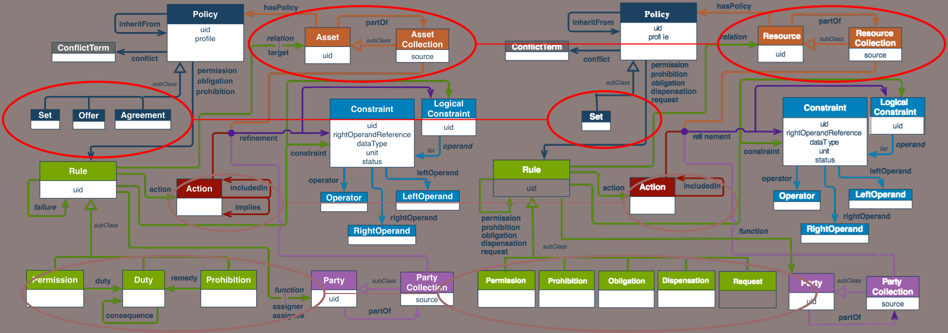
# ODRL information model → Policy Compliance profile



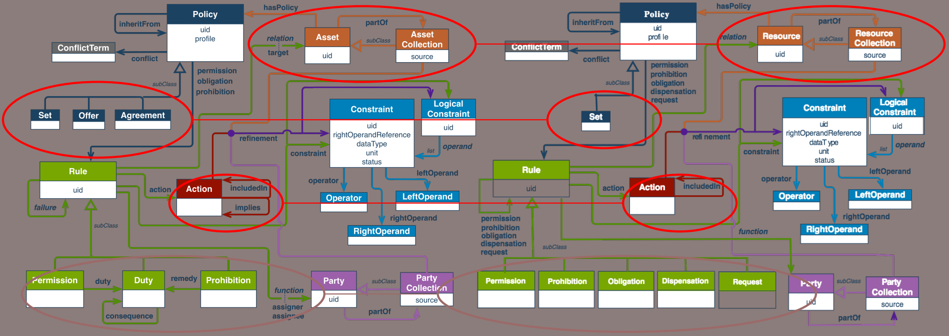
# ODRL information model → Policy Compliance profile



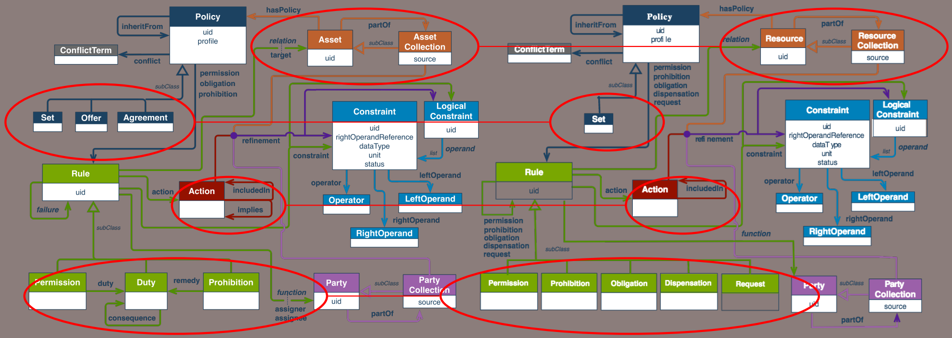
# ODRL information model → Policy Compliance profile



# ODRL information model → Policy Compliance profile



# ODRL information model → Policy Compliance profile



# Example ODRL policy compliance request

```
{
    "@context": "http://www.w3.org/ns/orcp.jsonld",
    "@type": "Set",
    "uid": "http://example.com/policy:01",
    "profile": "http://example.com/odrl:profile:regulatory-
        compliance",

    "request": [{
        "action": "orcp:Transfer",
        "target": "orcp:PersonalData",
        "sender": "http://example.com/TR_Ireland",
        "recipient": "http://example.com/TR_USA",
        "purpose": "orcp:KYC",
        "location": "orcp:USA",
        "legalBasis": "orcp:Consent",
        "constraint": [{
            "leftOperand": "orcp:AppropriateSafeguards",
            "operator": "eq",
            "rightOperand": { "@id": "orcp:BindingCorporateRules
                "  }
        }]
    }]
}
```

## Data protection

- use-case: fragments of articles of GDPR
- check business process compliance with GDPR
- H2020 SPECIAL project
- develop ODRL → InstAL translator
- aim to synthesize ODRL from natural language policies

## Data protection

- use-case: fragments of articles of GDPR
- check business process compliance with GDPR
- H2020 SPECIAL project
- develop ODRL → InstAL translator
- aim to synthesize ODRL from natural language policies

## Data protection

- use-case: fragments of articles of GDPR
- check business process compliance with GDPR
- H2020 SPECIAL project
- develop ODRL → InstAL translator
- aim to synthesize ODRL from natural language policies

## Data protection

- use-case: fragments of articles of GDPR
- check business process compliance with GDPR
- H2020 SPECIAL project
- develop ODRL → InstAL translator
- aim to synthesize ODRL from natural language policies

## Data protection

- use-case: fragments of articles of GDPR
- check business process compliance with GDPR
- H2020 SPECIAL project
- develop ODRL $\rightarrow$ InstAL translator
- aim to synthesize ODRL from natural language policies

# Contents

## Summary

- formal model of institutions: policies, ...
- (formal model of (directed) bridges: connect institutions)
- computational model
- achieves:
  - refactoring and decoupling of normative reasoning
  - publication of normative reasoning as a service
- enables/facilitates:
  - testing of normative models
  - sharing/re-use of normative models
  - certification of normative models (of regulation)
  - regulatory compliance-checking as a service

## Summary

- formal model of institutions: policies, ...
- (formal model of (directed) bridges: connect institutions)
- computational model
- achieves:
  - refactoring and decoupling of normative reasoning
  - publication of normative reasoning as a service
- enables/facilitates:
  - testing of normative models
  - sharing/re-use of normative models
  - certification of normative models (of regulation)
  - regulatory compliance-checking as a service

## Summary

- formal model of institutions: policies, ...
- (formal model of (directed) bridges: connect institutions)
- computational model
- achieves:
  - refactoring and decoupling of normative reasoning
  - publication of normative reasoning as a service
- enables/facilitates:
  - testing of normative models
  - sharing/re-use of normative models
  - certification of normative models (of regulation)
  - regulatory compliance-checking as a service

## Summary

- formal model of institutions: policies, ...
- (formal model of (directed) bridges: connect institutions)
- computational model
- achieves:
  - refactoring and decoupling of normative reasoning
  - publication of normative reasoning as a service
- enables/facilitates:
  - testing of normative models
  - sharing/re-use of normative models
  - certification of normative models (of regulation)
  - regulatory compliance-checking as a service

## Summary

- formal model of institutions: policies, ...
- (formal model of (directed) bridges: connect institutions)
- computational model
- achieves:
  - refactoring and decoupling of normative reasoning
  - publication of normative reasoning as a service
- enables/facilitates:
  - testing of normative models
  - sharing/re-use of normative models
  - certification of normative models (of regulation)
  - regulatory compliance-checking as a service

## Summary

- formal model of institutions: policies, ...
- (formal model of (directed) bridges: connect institutions)
- computational model
- achieves:
  - refactoring and decoupling of normative reasoning
  - publication of normative reasoning as a service
- enables/facilitates:
  - testing of normative models
  - sharing/re-use of normative models
  - certification of normative models (of regulation)
  - regulatory compliance-checking as a service

## Summary

- formal model of institutions: policies, ...
- (formal model of (directed) bridges: connect institutions)
- computational model
- achieves:
  - refactoring and decoupling of normative reasoning
  - publication of normative reasoning as a service
- enables/facilitates:
  - testing of normative models
  - sharing/re-use of normative models
  - certification of normative models (of regulation)
  - regulatory compliance-checking as a service

## Summary

- formal model of institutions: policies, ...
- (formal model of (directed) bridges: connect institutions)
- computational model
- achieves:
    - refactoring and decoupling of normative reasoning
    - publication of normative reasoning as a service
- enables/facilitates:
    - testing of normative models
    - sharing/re-use of normative models
    - certification of normative models (of regulation)
    - regulatory compliance-checking as a service

## Summary

- formal model of institutions: policies, ...
- (formal model of (directed) bridges: connect institutions)
- computational model
- achieves:
  - refactoring and decoupling of normative reasoning
  - publication of normative reasoning as a service
- enables/facilitates:
  - testing of normative models
  - sharing/re-use of normative models
  - certification of normative models (of regulation)
  - regulatory compliance-checking as a service

## Discussion

- how to share and test policy models?
- how to certify policy models?
- how to discover policy models for re-use?
- how to record policy states for audit?
- how to capture written policy formally?
- how to capture policy heterarchies?
- how to revise a policy: which is the master copy?

Acknowledgements: Marina De Vos[1], Sabrina Kirrane[2], Pablo Noriega[3], Charlie Ann Page[1], Ken Satoh[4], Harko Verhagen[5]

QUESTIONS WELCOME!

---

[1]University of Bath
[2]Vienna Business University
[3]IIIA/CSIC
[4]National Institute of Informatics, Tokyo
[5]Stockholm University

## Discussion

- how to share and test policy models?
- how to certify policy models?
- how to discover policy models for re-use?
- how to record policy states for audit?
- how to capture written policy formally?
- how to capture policy heterarchies?
- how to revise a policy: which is the master copy?

Acknowledgements: Marina De Vos[1], Sabrina Kirrane[2], Pablo Noriega[3], Charlie Ann Page[1], Ken Satoh[4], Harko Verhagen[5]

QUESTIONS WELCOME!

[1]University of Bath
[2]Vienna Business University
[3]IIIA/CSIC
[4]National Institute of Informatics, Tokyo
[5]Stockholm University

## Discussion

- how to share and test policy models?
- how to certify policy models?
- how to discover policy models for re-use?
- how to record policy states for audit?
- how to capture written policy formally?
- how to capture policy heterarchies?
- how to revise a policy: which is the master copy?

Acknowledgements: Marina De Vos[1], Sabrina Kirrane[2], Pablo Noriega[3], Charlie Ann Page[1], Ken Satoh[4], Harko Verhagen[5]

QUESTIONS WELCOME!

[1] University of Bath
[2] Vienna Business University
[3] IIIA/CSIC
[4] National Institute of Informatics, Tokyo
[5] Stockholm University

## Discussion

- how to share and test policy models?
- how to certify policy models?
- how to discover policy models for re-use?
- how to record policy states for audit?
- how to capture written policy formally?
- how to capture policy heterarchies?
- how to revise a policy: which is the master copy?

Acknowledgements: Marina De Vos[1], Sabrina Kirrane[2], Pablo Noriega[3], Charlie Ann Page[1], Ken Satoh[4], Harko Verhagen[5]

QUESTIONS WELCOME!

[1]University of Bath
[2]Vienna Business University
[3]IIIA/CSIC
[4]National Institute of Informatics, Tokyo
[5]Stockholm University

## Discussion

- how to share and test policy models?
- how to certify policy models?
- how to discover policy models for re-use?
- how to record policy states for audit?
- how to capture written policy formally?
- how to capture policy heterarchies?
- how to revise a policy: which is the master copy?

Acknowledgements: Marina De Vos[1], Sabrina Kirrane[2], Pablo Noriega[3], Charlie Ann Page[1], Ken Satoh[4], Harko Verhagen[5]

QUESTIONS WELCOME!

---

[1]University of Bath
[2]Vienna Business University
[3]IIIA/CSIC
[4]National Institute of Informatics, Tokyo
[5]Stockholm University

## Discussion

- how to share and test policy models?
- how to certify policy models?
- how to discover policy models for re-use?
- how to record policy states for audit?
- how to capture written policy formally?
- how to capture policy heterarchies?
- how to revise a policy: which is the master copy?

Acknowledgements: Marina De Vos[1], Sabrina Kirrane[2], Pablo Noriega[3], Charlie Ann Page[1], Ken Satoh[4], Harko Verhagen[5]

QUESTIONS WELCOME!

[1]University of Bath
[2]Vienna Business University
[3]IIIA/CSIC
[4]National Institute of Informatics, Tokyo
[5]Stockholm University

## Discussion

- how to share and test policy models?
- how to certify policy models?
- how to discover policy models for re-use?
- how to record policy states for audit?
- how to capture written policy formally?
- how to capture policy heterarchies?
- how to revise a policy: which is the master copy?

Acknowledgements: Marina De Vos[1], Sabrina Kirrane[2], Pablo Noriega[3], Charlie Ann Page[1], Ken Satoh[4], Harko Verhagen[5]

QUESTIONS WELCOME!

[1]University of Bath
[2]Vienna Business University
[3]IIIA/CSIC
[4]National Institute of Informatics, Tokyo
[5]Stockholm University

## Discussion

- how to share and test policy models?
- how to certify policy models?
- how to discover policy models for re-use?
- how to record policy states for audit?
- how to capture written policy formally?
- how to capture policy heterarchies?
- how to revise a policy: which is the master copy?

Acknowledgements: Marina De Vos[1], Sabrina Kirrane[2], Pablo Noriega[3], Charlie Ann Page[1], Ken Satoh[4], Harko Verhagen[5]

QUESTIONS WELCOME!

[1] University of Bath
[2] Vienna Business University
[3] IIIA/CSIC
[4] National Institute of Informatics, Tokyo
[5] Stockholm University

# Discussion

- how to share and test policy models?
- how to certify policy models?
- how to discover policy models for re-use?
- how to record policy states for audit?
- how to capture written policy formally?
- how to capture policy heterarchies?
- how to revise a policy: which is the master copy?

Acknowledgements: Marina De Vos[1], Sabrina Kirrane[2], Pablo Noriega[3], Charlie Ann Page[1], Ken Satoh[4], Harko Verhagen[5]

QUESTIONS WELCOME!

[1] University of Bath
[2] Vienna Business University
[3] IIIA/CSIC
[4] National Institute of Informatics, Tokyo
[5] Stockholm University

# Bibliography I


Aldewereld, Huib, Olivier Boissier, Virginia Dignum, Pablo Noriega, and Julian Padget, eds. (2016). *Social Coordination Frameworks for Social Technical Systems*. Vol. 30. Law, Governance and Technology. Springer. DOI: 10.1007/978-3-319-33570-4_.

Behrens, Tristan, Koen Hindriks, and Jürgen Dix (2011). "Towards an environment interface standard for agent platforms". In: *Ann. Math. Artif. Intell.* 61.4, pp. 261–295. DOI: 10.1007/s10472-010-9215-9.

Bordini, Rafael, Jomi Hübner, and Michael Wooldridge (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, Ltd. DOI: 10.1002/9780470061848.ch4.

Harré, Rom and Paul F Secord (1972). "The explanation of social behaviour.". In:

Jones, A.J.I. and M. Sergot (1996). "A formal characterisation of institutionalised power". In: *Logic Journal of IGPL* 4.3, pp. 427–443.

# Bibliography II


- Lee, Jeehang, Julian Padget, Brian Logan, Daniela Dybalova, and Natasha Alechina (2014). "N-Jason: Run-Time Norm Compliance in AgentSpeak(L)". In: *Engineering Multi-Agent Systems - Second International Workshop, EMAS 2014, Paris, France, May 5-6, 2014, Revised Selected Papers*. Ed. by Fabiano Dalpiaz, Jürgen Dix, and M. Birna van Riemsdijk. Vol. 8758. Lecture Notes in Computer Science. Springer, pp. 367–387. DOI: 10.1007/978-3-319-14484-9_19.
- Noriega, Pablo, Jordi Sabater-Mir, Harko Verhagen, Julian Padget, and Mark d'Inverno (2017). "Identifying Affordances for Modelling Second-Order Emergent Phenomena with the *WIT* Framework". In: *Autonomous Agents and Multiagent Systems - AAMAS 2017 Workshops, Visionary Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers*. Ed. by Gita Sukthankar and Juan A. Rodriguez-Aguilar. Vol. 10643. Lecture Notes in Computer Science. Springer, pp. 208–227. DOI: 10.1007/978-3-319-71679-4_14.
- North, Douglas (1991). *Institutions, Institutional Change and Economic Performance*. CUP.
- Ostrom, Elinor (1990). *Governing the Commons. The Evolution of Institutions for Collective Action*. CUP.

# Bibliography III


Padget, Julian, Emad Elakehal, Tingting Li, and Marina De Vos (2016a). "InstAL: An Institutional Action Language". In: *Social Coordination Frameworks for Social Technical Systems*. Springer, pp. 101–124.

Padget, Julian, Emad ElDeen Elakehal, Tingting Li, and Marina De Vos (2016b). "InstAL: An Institutional Action Language". In: *Social Coordination Frameworks for Social Technical Systems*. Springer, pp. 101–124. DOI: 10.1007/978-3-319-33570-4_6.

Padget, Julian, Marina De Vos, and Charlie Ann Page (July 2018). "Deontic Sensors". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, pp. 475–481. DOI: 10.24963/ijcai.2018/66.

Searle, John (1995). *The Construction of Social Reality*. Allen Lane, The Penguin Press.

Wright, Georg von (1951). "Deontic Logic". In: *Mind* 60.237, pp. 1–15.