Representing and reasoning about policy for agent-based simulation

Julian Padget

Department of Computer Science, University of Bath

April 2, 2019
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)
  - NJJason: 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)
  - NJJason: 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)
  - NJJason: 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 $\equiv$ policies $\equiv$ regulations $\equiv$ narratives $\equiv$ 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)
- NJJason: 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)
- NJJason: 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)
  - NJJason: 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:
  - 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:
  - 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:
  - 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
Contents

1 Introduction

2 InstAL: a DSL for norm modelling

3 Deontic Sensors: normative reasoning as a service

4 Sample water management policy

5 Semantic policy representation

6 Epilogue
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
- $\Rightarrow$ knowledge representation as norms
- $\Rightarrow$ governance of agents in multiagent systems
- $\Rightarrow$ 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

- ↳ 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

- → knowledge representation as norms
- → governance of agents in multiagent systems
- → 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 $\rightarrow$ social facts
- counts-as (Jones et al. 1996):
  - real-world event $\rightarrow$ 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 $\Rightarrow$ social facts

- counts-as (Jones et al. 1996):
  - real-world event $\Rightarrow$ institutional event

Human/Software Actors \[\text{brute social}\] Deontic Sensor
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 $\leadsto$ social facts

- counts-as (Jones et al. 1996):
  - real-world event $\leadsto$ 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)
- inspiration: social sciences (Harré et al. 1972)
- inspiration: 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

Human/Software Actors $\rightarrow$ Deontic Sensor

brute $\leftrightarrow$ social
Actions change the (institutional) world

- counts-as: $G : \text{external action} \xrightarrow{\text{generates}} \text{institutional action}$

- institutional facts represented by fluents
  1. fluent $\Rightarrow$ true if present, false otherwise
  2. inertial fluent:
  $C^\uparrow : \text{action} \xrightarrow{\text{initiates}} \text{fluent}$
  $C^\downarrow : \text{action} \xrightarrow{\text{terminates}} \text{fluent}$
  good for facts true for a period with start and finish actions
  3. non-inertial fluent:
  $C^N : \text{fluent} \xrightarrow{\text{if(conditions)}}$
  good for facts expressed as combination of inertial + non-inertial fluents
Actions change the (institutional) world

- counts-as: \( G \) : external action \( \iff \) (conditions) \( \rightarrow \) institutional action

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

  \( C^\uparrow \): action \( \iff \) (conditions) \( \rightarrow \) fluent

  \( C^\downarrow \): action \( \iff \) (conditions) \( \rightarrow \) fluent

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

  \( C^\mathcal{N} \): action \( \iff \) (conditions) \( \rightarrow \) fluent

  good for facts expressed as combination of inertial + non-inertial fluents
Actions change the (institutional) world

- counts-as: \( G \) : external action \( \text{generates} \) institutional action

- institutional facts represented by fluents
  1. fluent \( \Rightarrow \) true if present, false otherwise
  2. inertial fluent:
     \( C^\uparrow \): action \( \text{initiates} \) fluent
     \( C^\downarrow \): action \( \text{terminates} \) fluent
     good for facts true for a period with start and finish actions
  3. non-inertial fluent:
     \( C^N \) : fluent
     good for facts expressed as combination of inertial + non-inertial fluents
Actions change the (institutional) world

- **counts-as:** \( G \) : external action \( \Rightarrow \) if(conditions) \( \Rightarrow \) institutional action

- **institutional facts represented by fluents**
  1. fluent \( \Rightarrow \) true if present, false otherwise
  2. inertial fluent:

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

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

3. non-inertial fluent:

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

\[
C^N: \text{fluent} \quad \text{if(conditions)}
\]

- **good for facts expressed as combination of inertial + non-inertial fluents**
Actions change the (institutional) world

- **counts-as:** \(G\) : external action \(\Rightarrow\) generates institutional action if (conditions)

- **institutional facts represented by fluents**
  1. fluent \(\Rightarrow\) true if present, false otherwise
  2. inertial fluent:
  \(C^\uparrow\) : action \(\Rightarrow\) initiates fluent if (conditions)
  \(C^\downarrow\) : action \(\Rightarrow\) terminates fluent if (conditions)

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

- **non-inertial fluent:** \(C^N\) : fluent if (conditions)
  good for facts expressed as combination of inertial + non-inertial fluents
Making it work

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

\[
\Delta \xrightarrow{e_1} S_1 \xrightarrow{e_2} S_2 \xrightarrow{e_3} \cdots
\]

- conceptual model:
  generalizes to multiple, connected institutions
Making it work

- **Mathematical model:**
  
  \[
  \Delta \xrightarrow{e_1} S_1 \xrightarrow{e_2} S_2 \xrightarrow{e_3} \ldots
  \]

- **Conceptual model:**

  Generalizes to multiple, connected institutions
Making it work

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

\[
\Delta \xrightarrow{e_1} S_1 \xrightarrow{e_2} S_2 \xrightarrow{e_3} ... 
\]

- conceptual model:

- generalizes to multiple, connected institutions
Implementation

- computational model:
  \[\text{InstAL} \xrightarrow{\text{translator}} \text{AnsProlog} \xrightarrow{\text{clingo}} \text{answer sets} \xrightarrow{\text{visualisers}}\]

- 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 → 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 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
Contents

1. Introduction

2. InstAL: a DSL for norm modelling

3. Deontic Sensors: normative reasoning as a service

4. Sample water management policy

5. Semantic policy representation

6. Epilogue
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
  \( \Rightarrow \) decoupling
  \( \Rightarrow \) institution re-use
  \( \Rightarrow \) 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
- \( \rightarrow \) decoupling
- \( \rightarrow \) institution re-use
- \( \rightarrow \) 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
  - $\Rightarrow$ decoupling
  - $\Rightarrow$ institution re-use
  - $\Rightarrow$ 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
- \( \Rightarrow \) decoupling
- \( \Rightarrow \) institution re-use
- \( \Rightarrow \) 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
- $\mapsto$ decoupling
- $\mapsto$ institution re-use
- $\mapsto$ 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
  $\Rightarrow$ decoupling
  $\Rightarrow$ institution re-use
  $\Rightarrow$ 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
- $\rightarrow$ decoupling
- $\rightarrow$ institution re-use
- $\rightarrow$ 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. \( \rightarrow \) POST /model/
   
   Create model from specification

   \( \leftarrow \) /model/X

2. \( \rightarrow \) POST /model/X/instance/
   
   Create instance of model X with POST data

   \( \leftarrow \) /model/X/instance/Y

3. \( \rightarrow \) POST /model/X/instance/Y/query/
   
   Create a query of instance Y with POST data

   \( \leftarrow \) /model/X/instantiate/Y/query/Z

4. \( \rightarrow \) GET /model/X/instance/Y/query/Z/output
   
   Read result of query

   \( \leftarrow \) 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: Institutional Action Language
  - 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: Institutional Action Language
  - 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: Institutional Action Language
  - 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: **Institutional Action Language**
  - 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: **Institutional Action Language**
  - 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: Institutional Action Language
  - 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: **Institutional Action Language**
  - 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

1 Introduction

2 InstAL: a DSL for norm modelling

3 Deontic Sensors: normative reasoning as a service

4 Sample water management policy

5 Semantic policy representation

6 Epilogue
InstAL declarations

Different types of declarations:

1. type Industry;
2. exogenous event discharge(WWTP,Mass);
3. violation event illegalDischarge(WWTP,Mass);
4. inst event iDischarge(WWTP,Mass);
5. fluent highMercury(Mass);
6. obligation fluent obl(
   7. iInform(Industry,WWTP,Mass),               % event
   8. iRelease(Industry,WWTP,Mass),             % deadline
   9. failureToInform(Industry,WWTP,Mass));     % violation
InstAL rules

Generates, initiates and terminates rules:

1. discharge(WWTP,Mass) generates iDischarge(WWTP,Mass)
2. if treated(WWTP,Mass,Treatment);
3. discharge(WWTP,Mass) generates illegalDischarge(WWTP,Mass)
4. if not treated(WWTP,Mass,Treatment);
5. discharge(WWTP,Mass) generates illegalDischarge(WWTP,Mass)
6. if highMercury(Mass);
7. illegalDischarge(WWTP,Mass) initiates illegalBecause(untreated,WWTP,Mass)
8. if not treated(WWTP,Mass,Treatment);
9. illegalDischarge(WWTP,Mass) initiates illegalBecause(high_mercury,WWTP,Mass)
10. if highMercury(Mass);
11. iDischarge(WWTP,Mass) terminates treated(WWTP,Mass,Treatment)
12. if treated(WWTP,Mass,Treatment), not highMercury(Mass);
13. iPerform(WWTP,Mass,Treatment) initiates treated(WWTP,Mass,Treatment)
14. if treating(WWTP,Mass);
15. initially
16. highMercury(m2),
17. signedContract(wwtp1,i1),
18. obl(iInform(i1,wwtp1,M),iRelease(i1,wwtp1,M),failureToInform(i1,wwtp1,M))
Sample run

- **grounding specification:**
  
  1. **Industry:** i1 i2  
  2. **Mass:** m1 m2  
  3. **Reason:** untreated high_mercury  
  4. **Treatment:** tk  
  5. **WWTP:** wwtp1 wwtp2

- **input trace:**
  
  1. observed(inform(i1,wwtp1,m2))  
  2. observed(release(i1,wwtp1,m2))  
  3. observed(receive(wwtp1,i1,m2))  
  4. observed(perform(wwtp1,m2,tk))  
  5. observed(discharge(wwtp1,m2))  
  6. observed(release(i2,wwtp2,m1))  
  7. observed(receive(wwtp2,i2,m1))  
  8. observed(discharge(wwtp2,m1))
Sample run

- **grounding specification:**
  1. Industry: i1 i2
  2. Mass: m1 m2
  3. Reason: untreated high_mercury
  4. Treatment: tk
  5. WWTP: wwtp1 wwtp2

- **input trace:**
  1. observed(inform(i1,wwtp1,m2))
  2. observed(release(i1,wwtp1,m2))
  3. observed(receive(wwtp1,i1,m2))
  4. observed(perform(wwtp1,m2,tk))
  5. observed(discharge(wwtp1,m2))
  6. observed(release(i2,wwtp2,m1))
  7. observed(receive(wwtp2,i2,m1))
  8. observed(discharge(wwtp2,m1))
Trace visualization
Contents

1 Introduction

2 InstAL: a DSL for norm modelling

3 Deontic Sensors: normative reasoning as a service

4 Sample water management policy

5 Semantic policy representation

6 Epilogue
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 → 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 → 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 → 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
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

```json
{
  "@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 $\rightarrow$ 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
Contents

1. Introduction

2. InstAL: a DSL for norm modelling

3. Deontic Sensors: normative reasoning as a service

4. Sample water management policy

5. Semantic policy representation

6. Epilogue
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 Vos¹, Sabrina Kirrane², Pablo Noriega³, Charlie Ann Page¹, Ken Satoh⁴, Harko Verhagen⁵

QUESTIONS WELCOME!

¹University of Bath
²Vienna Business University
³IIIA/CSIC
⁴National Institute of Informatics, Tokyo
⁵Stockholm 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 Vos¹, Sabrina Kirrane², Pablo Noriega³, Charlie Ann Page¹, Ken Satoh⁴, Harko Verhagen⁵

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 Vos¹, Sabrina Kirrane², Pablo Noriega³, Charlie Ann Page¹, Ken Satoh⁴, Harko Verhagen⁵

QUESTIONS WELCOME!

¹University of Bath
²Vienna Business University
³IIIA/CSIC
⁴National Institute of Informatics, Tokyo
⁵Stockholm 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\(^5\)

\(^1\)University of Bath
\(^2\)Vienna Business University
\(^3\)IIIA/CSIC
\(^4\)National Institute of Informatics, Tokyo
\(^5\)Stockholm University

\textbf{QUESTIONS WELCOME!}
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\textsuperscript{1}, Sabrina Kirrane\textsuperscript{2}, Pablo Noriega\textsuperscript{3}, Charlie Ann Page\textsuperscript{1}, Ken Satoh\textsuperscript{4}, Harko Verhagen\textsuperscript{5}

QUESTIONS WELCOME!

\textsuperscript{1}University of Bath
\textsuperscript{2}Vienna Business University
\textsuperscript{3}IIIA/CSIC
\textsuperscript{4}National Institute of Informatics, Tokyo
\textsuperscript{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 Vos¹, Sabrina Kirrane², Pablo Noriega³, Charlie Ann Page¹, Ken Satoh⁴, Harko Verhagen⁵

QUESTIONS WELCOME!

¹University of Bath
²Vienna Business University
³IIIA/CSIC
⁴National Institute of Informatics, Tokyo
⁵Stockholm 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\textsuperscript{1}, Sabrina Kirrane\textsuperscript{2}, Pablo Noriega\textsuperscript{3}, Charlie Ann Page\textsuperscript{1}, Ken Satoh\textsuperscript{4}, Harko Verhagen\textsuperscript{5}

\textbf{QUESTIONS WELCOME!}

\textsuperscript{1}University of Bath
\textsuperscript{2}Vienna Business University
\textsuperscript{3}IIIA/CSIC
\textsuperscript{4}National Institute of Informatics, Tokyo
\textsuperscript{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\textsuperscript{1}, Sabrina Kirrane\textsuperscript{2}, Pablo Noriega\textsuperscript{3}, Charlie Ann Page\textsuperscript{1}, Ken Satoh\textsuperscript{4}, Harko Verhagen\textsuperscript{5}

\textbf{QUESTIONS WELCOME!}

\textsuperscript{1}University of Bath
\textsuperscript{2}Vienna Business University
\textsuperscript{3}IIIA/CSIC
\textsuperscript{4}National Institute of Informatics, Tokyo
\textsuperscript{5}Stockholm University
Bibliography I


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


