

Normative System Programming

John-Jules Meyer

*Joint work with Mehdi Dastani, Nick
Tinnemeier, Davide Grossi*



Universiteit Utrecht

Main idea

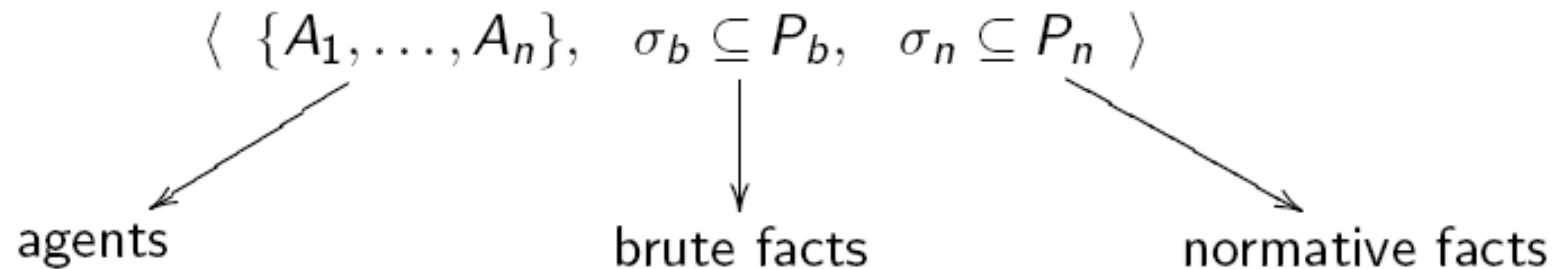
- Design and develop a *programming language* to support the implementation of coordination mechanisms in terms of normative concepts.
 - I (literally ;-) show you some snapshots from this work.
-

Normative programming language

Programming a normative multi-agent organization is to specify:

- ▶ references to *2APL agent programs*, e.g.,
`passenger PassProg 1`
 - ▶ the initial state of its environment by *brute facts*, e.g.,
`{-at_platform, -in_train, -ticket}`
 - ▶ the *effects of actions* in the environment, e.g.,
`{-ticket} buy_ticket {ticket}`
`{at_platform, -in_train} embark {-at_platform, in_train}`
 - ▶ the norms through *counts-as rules*, e.g.,
`{at_platform , -ticket} ⇒ {viol1}`
`{in_train , -ticket} ⇒ {viol1}`
 - ▶ possible sanctions for agent's through *sanction rules*, e.g.,
`{viol1} ⇒ {fined10}`
-

Semantics



- ▶ P_b and P_n are disjoint sets of literals
 - ▶ Brute facts describe current state of the environment
 - ▶ Normative facts describe normative assessment of organisation
-

Semantics (ctd)

When an agent performs an external action, the organization:

- ▶ determines new brute state based on effect rules
 - by using function $up(\alpha(i), \sigma_b)$, i.e. $\sigma'_b = up(\alpha(i), \sigma_b)$
- ▶ normatively judges this state by applying counts-as rules
 - by taking the closure of σ'_b under rules R_c , i.e. $\sigma'_n = cl^{R_c}(\sigma'_b) \setminus \sigma'_b$

Based on this normative judgment, the organization either:

- ▶ effectuates action and applies all sanction rules accordingly
 - by taking closure of σ'_n under rules R_s , i.e. $S = cl^{R_s}(\sigma'_n) \setminus \sigma'_n$
 - new brute state becomes $\sigma'_b \cup S$
 - ▶ blocks the action if it would lead to a state marked by $viol_{\perp}$
-

Semantics (ctd)

- ▶ Applicable rules given set of literals X and rules \mathbf{R} , $\text{App1}^{\mathbf{R}}(X)$:

$$\text{App1}^{\mathbf{R}}(X) = \{\Phi \Rightarrow \Psi \mid X \models \Phi\}$$

- ▶ Closure of set of literals X under rules \mathbf{R} , $\text{cl}^{\mathbf{R}}(X)$:

$$\text{cl}^{\mathbf{R}}(X) = \text{cl}_{m+1}^{\mathbf{R}}(X) \text{ iff } \text{cl}_{m+1}^{\mathbf{R}}(X) = \text{cl}_m^{\mathbf{R}}(X), \text{ where}$$

$$\text{B: } \text{cl}_0^{\mathbf{R}}(X) = X \cup \left(\bigcup_{l \in \text{App1}^{\mathbf{R}}(X)} \text{cons}_l \right)$$

$$\text{S: } \text{cl}_{n+1}^{\mathbf{R}}(X) = \text{cl}_n^{\mathbf{R}}(X) \cup \left(\bigcup_{l \in \text{App1}^{\mathbf{R}}(\text{cl}_n^{\mathbf{R}}(X))} \text{cons}_l \right)$$

- ▶ Effect of action $\alpha(i)$ specified by brute effect rule $(\Phi \alpha(i) \Phi')$:

$$\text{up}(\alpha(i), \sigma_b) = (\sigma_b \cup \Phi') \setminus (\{p \mid -p \in \Phi'\} \cup \{-p \mid p \in \Phi'\})$$

Let action $\alpha(i)$ be specified as: $(\Phi \ \alpha(i) \ \Phi')$.

Transition rule for individual agent's external actions:

$A_i \xrightarrow{\alpha(i)} A'_i$: agent i can perform external action α .

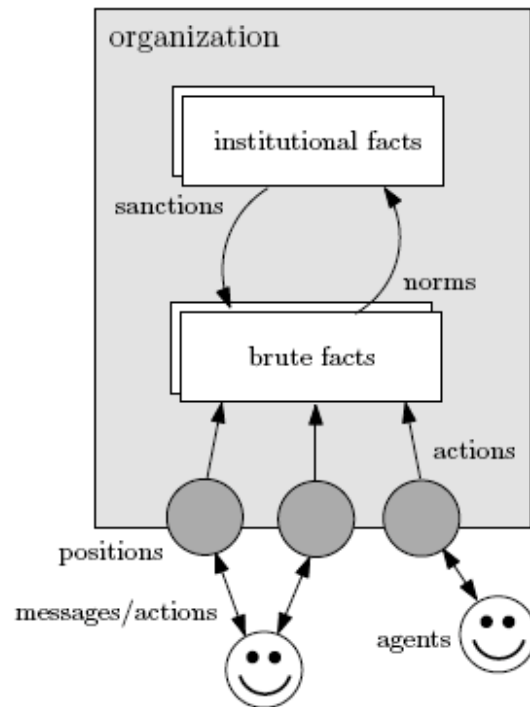
Transition rule for normative multi-agent organization:

$$\frac{A_i \in \mathcal{A} \quad A_i \xrightarrow{\alpha(i)} A'_i \quad \sigma_b \models \Phi \quad \sigma'_b = up(\alpha(i), \sigma_b) \quad \sigma'_n = \text{Cl}^{\text{Rc}}(\sigma'_b) \setminus \sigma'_b \quad \sigma'_n \not\models \text{viol}_\perp \quad S = \text{Cl}^{\text{Rs}}(\sigma'_n) \setminus \sigma'_n \quad \sigma'_b \cup S \not\models \perp}{\langle \mathcal{A}, \sigma_b, \sigma_n \rangle \longrightarrow \langle \mathcal{A}', \sigma'_b \cup S, \sigma'_n \rangle}$$

where $\mathcal{A}' = (\mathcal{A} \setminus \{A_i\}) \cup \{A'_i\}$ and viol_\perp is the designated literal for regimentation.

Agent organizations: adding roles

An organization encapsulates a domain specific state and function that can be used by agents in achieving their goals.



- **Brute facts** model the domain specific state.
- Brute state is assessed by **norms**.
- **Institutional facts** model the normative judgment.
- Normative judgment might lead to **sanctions**.
- Agents interact with organization via **positions** (enacted roles).
- Positions can alter the brute state by performing **actions**.

Programming organizations in 2OPL

To program an organization is to specify the roles that can be played, the initial brute state, the effect actions have on the brute facts and the normative and sanctioning rules, e.g. a simplified implementation of a conference management system:

Roles:

```
reviewer: rev.rol  
author:  auth.rol
```

Brute Facts:

```
papers([]), reviews([]), invited(jj)
```

Effect Rules:

```
{invited(X)} register(X) {pc(X)}  
{papers(Y)} uploadP(X,PIId) {-papers(Y), papers([PIId|Y])}
```

Programming organizations (ctd)

Counts-as Rules:

$\{ \text{papers}(Ps), P \in Ps, \text{size}(P) > 15 \}$

\Rightarrow

$\{ \text{viol}(\text{size}, P, Ps) \}$

Sanction Rules:

$\{ \text{viol}(\text{size}, P, Ps) \}$

\Rightarrow

$\{ \text{not paper}(Ps), Ps' = \text{del}(P, Ps), \text{papers}(Ps') \}$

Programming roles

Roles are programmed in terms of BDI-concepts (beliefs, goals and plans). Consider for example an implementation of the reviewer role:

Beliefs:

```
reviews([])
```

Goals:

```
aidWithReview
```

Constructors:

```
enact(name(N), amount(Nr)) ← invited(N) & Nr ≥ 3 {  
  AddBelief(nr(Nr));  
  AddBelief(name(N));  
  @register(N); }  
}
```

PG-rules:

```
aidWithReview ← name(N) & assigned(N,X) {  
  request(review(X)); }  
}
```

Perceiving actions

Agents interact with the organization by performing actions. The organization is capable of perceiving these actions.

Transition Rule (Agent-Organization Interaction)

$$\frac{a \xrightarrow{\alpha!} a' \quad O \vdash \alpha? \rightarrow O'}{\langle A, O \rangle \Rightarrow \langle (A \setminus \{a\}) \cup \{a'\}, O' \rangle}$$

Transition Rule (Perceive Action)

$$\frac{}{\langle P, B, I, E \rangle \vdash \alpha? \rightarrow \langle P, B, I, E.\alpha \rangle}$$

Executing positions

Positions execute actions to change their own internals (e.g. beliefs and goals) and to modify the organization's brute facts.

Transition Rule (External Action)

Assume:

- $\rho = \langle p, \Sigma, \Gamma, \Pi \rangle$ and $\rho' = \langle p, \Sigma', \Gamma', \Pi' \rangle$
- $\text{up}(X, \delta)$ determine effect of action δ on facts X
- $\text{Cl}^R(X)$ computes closure of set of facts X under rules R

$$\frac{\rho \in P \quad \rho \xrightarrow{\delta!} \rho' \quad B' = \text{up}(B, \delta) \quad I' = \text{Cl}^C(B') \setminus (B') \quad B'' = \text{Cl}^S(I') \setminus (I')}{\langle P, B, I, E \rangle \mapsto \langle (P \setminus \{\rho\}) \cup \{\rho'\}, B' \cup B'', I', E \rangle}$$

Role enactment

Enacting a role is to find a matching constructor of which the precondition holds and executing it to create a position. An agent i enacts a role r by performing an $enact(i, r, \Phi)$ action.

Transition Rule (Role Enactment)

Assume:

- $enact(\Psi) \leftarrow \varphi \mid \pi$ is a constructor of role r

$$\frac{\text{unify}(\Phi, \Psi) = \tau_1 \quad (\Sigma, \Gamma, B, I) \models_t \varphi \tau_1 \tau_2}{\langle \langle p, \Sigma, \Gamma, \{\pi \tau_1 \tau_2\} \rangle, B, I, E \rangle \mapsto^* \langle \langle p, \Sigma', \Gamma', \{\} \rangle, B', I', E \rangle}$$
$$\langle P, B, I, enact(i, r, \Phi).E \rangle \xrightarrow{enacted(i, p)!} \langle P \cup \langle p, \Sigma', \Gamma', \{\} \rangle, B', I', E \rangle$$

Configuring positions

Once enacted players configure their positions by performing actions to alter and inspect the position's mental state.

Transition Rule (Add/Drop Goal)

Assume:

- $\rho = \langle p, \Sigma, \Gamma, \Pi \rangle$

$$\frac{\rho \in P}{\langle P, B, I, \alpha.E \rangle \mapsto \langle (P \setminus \{\rho\}) \cup \rho', B, I, E \rangle}$$

$$\text{where } \rho' = \begin{cases} \langle p, \Sigma, \Gamma \cup \{\phi\}, \Pi \rangle & \text{if } \alpha = \text{adoptg}(p, \phi) \\ \langle p, \Sigma, \Gamma \setminus \{\phi\}, \Pi \rangle & \text{if } \alpha = \text{dropg}(p, \phi) \end{cases}$$

Conclusion

- We work on extensions of this framework with:
 - richer notion of norms (conditional obligations with deadlines)
 - communication between roles
 - roles enacting roles
 - constructs for changing norms
 - logic to verify properties of normative multi-agent program.
-