Norm-based behaviour modification in BDI agents

Felipe Meneguzzi

felipe.meneguzzi@kcl.ac.uk

Michael Luck

michael.luck@kcl.ac.uk

Outline

- Context
- Norms in Agent Languages
- Normative Processing
- Normative AgentSpeak(L)
- Conclusions and Future Work

CONTEXT

Societal Control

- Multiagent systems:
  - Multiple autonomous agents
  - Adaptable to changes
  - Certain degree of unpredictability
  - Require societal control
- Norms provide a valuable mechanism to impose control on agent societies

Normative Systems

- Define standards of acceptable behaviour
- Rely on representation of:
  - Obligations
  - Prohibitions
  - Permissions
- Research largely on the *macro* level
- We need to address how individual agents *adapt* to norms, should they choose to follow them

NORMS IN AGENT LANGUAGES

Norm Representation

- Focuses on the operational aspect of norm compliance by an agent
- Norms are defined in the form:

  norm(Activation, Expiration, NormCondition)
- Denoting when a norm becomes active and expires, and what is the object of the norm

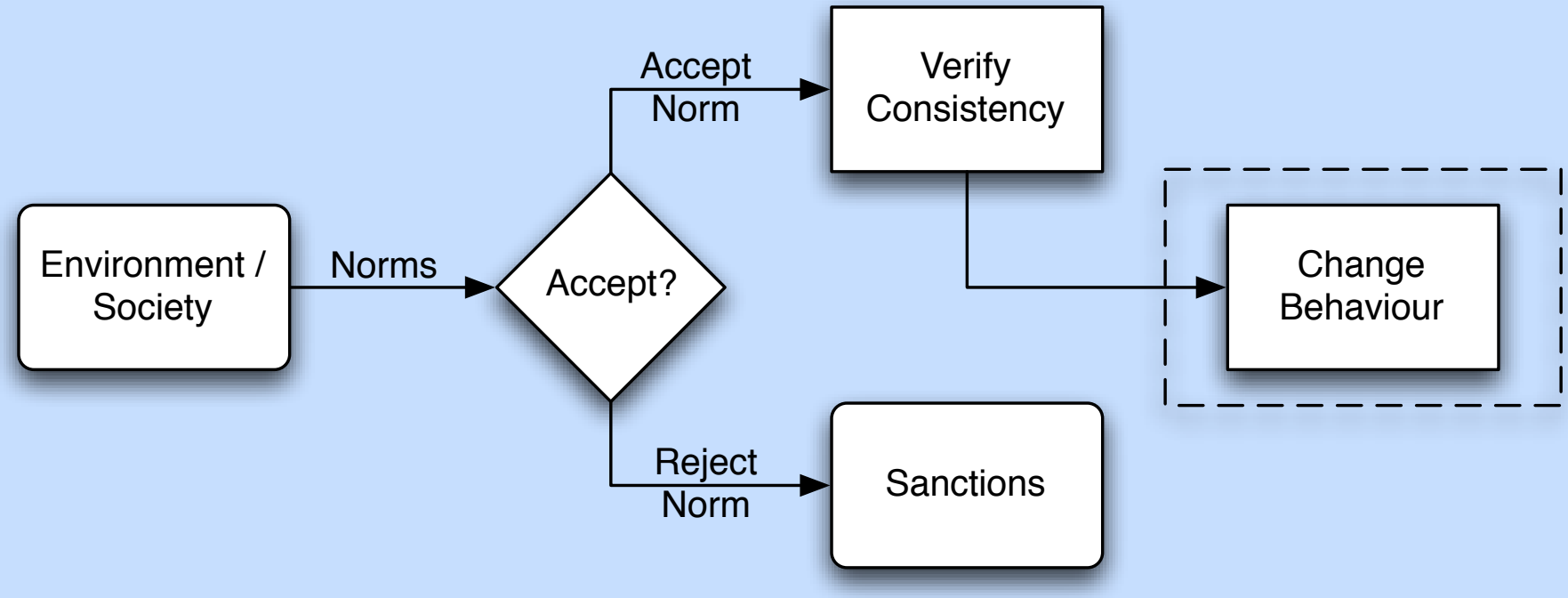
Norms and Goal Types

- We narrow norm types down to:
  - Obligations – agent *must* do/achieve something
  - Prohibitions – agent *must not* do/achieve something

Norm	Meaning
obligation(p)	add a goal to achieve state p, from *Activation* to *Expiration*.
obligation(a)	add a new plan with a *Activation* triggering event, and action a in its body.
prohibition(p)	prevent adoption of plans that bring about state p.
prohibition(a)	prevent adoption of plans that execute action a.

Norm Perception

- Norms perceived as environmental information
- If accepted, the following flow occurs



Motivating Example

Agent (in AgentSpeak)

```
+!cleanRoom(Room) : at(Room)
   <- +clean(Room).

+!clean(room1) : true
   <- +at(room1);
      !cleanRoom(room1).

+!clean(classifRoom) : true
   <- +at(classifRoom);
      !cleanRoom(classifRoom).

+cleanClassif : true
   <- !clean(classifRoom).
```

Norms

```
norm(time(4),
     time(20),
     obligation(clean(room1)))

norm(time(6),
     time(22),
   prohibition(at(classifRoom))
```

Expected Result

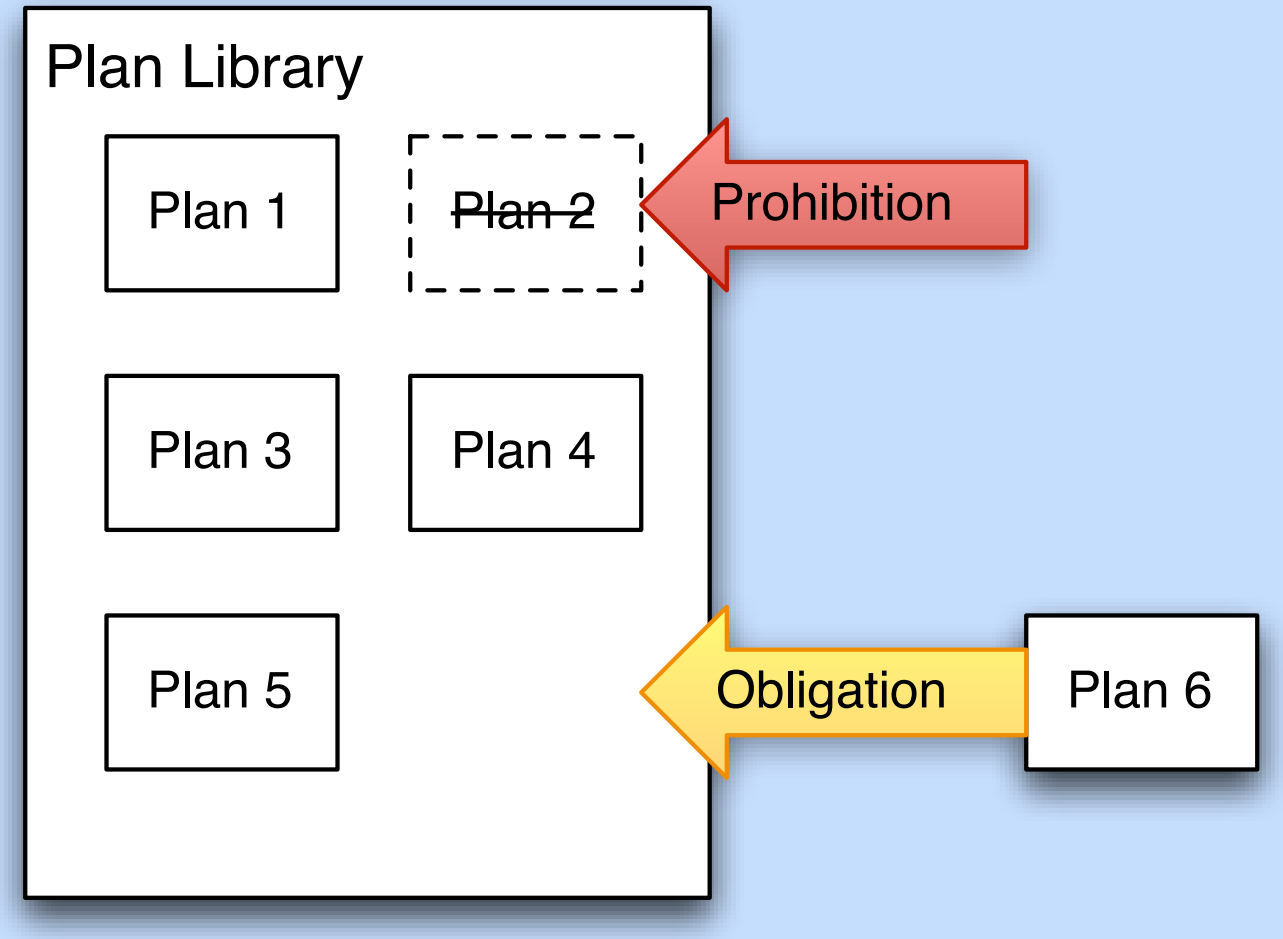
Events

- `time(4)`
- `time(6)`
- `cleanClassif`
- `time(20)`
- `time(22)`

Effects

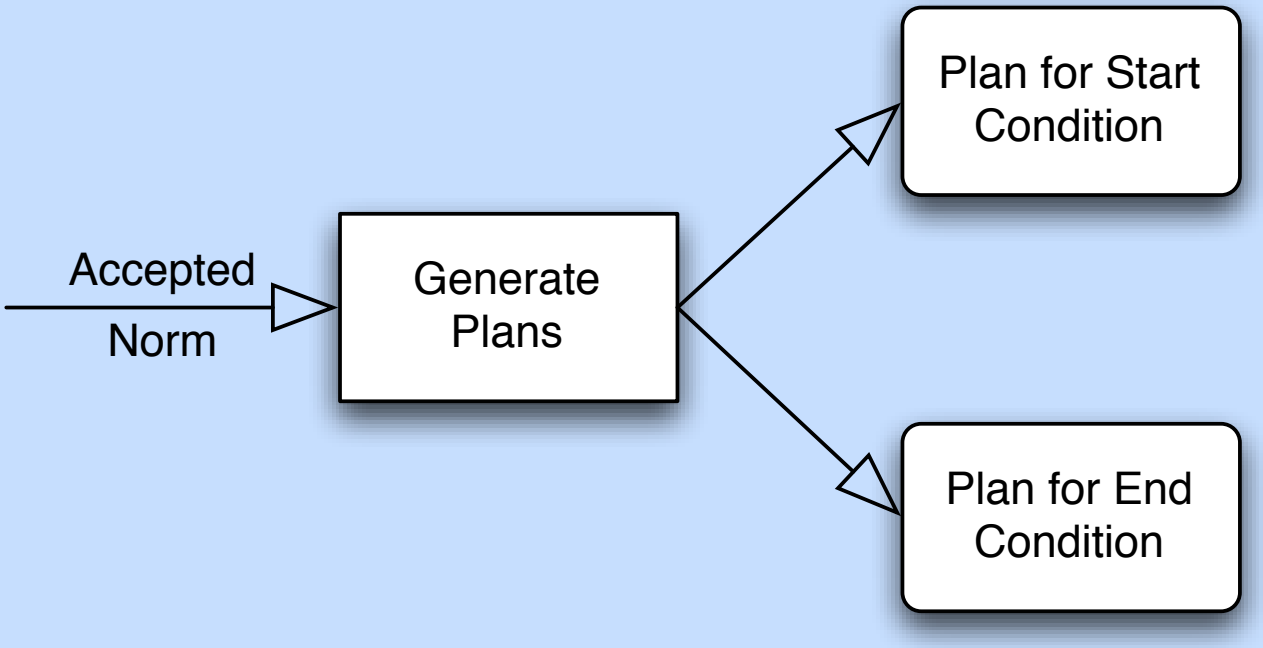
- Adopt plan to clean room1
- Suppress plan to clean classifRoom
- No plan should be adopted
- Obligation to clean room1 expires
- Plan to clean classifRoom no longer suppressed

In a nutshell



NORMATIVE PROCESSING

Norm Outcomes



Norm Activation

- Obligations
  - Behaviours associated with obligations must be carried out when they become active
  - Activation condition becomes trigger for plans that achieve obligations
- Prohibitions
  - Behaviours associated with prohibitions must not be carried out when they become active
  - Activation conditions becomes trigger for plans that *filter* intentions and plan library

Norm Expiration

- When a norm expires, its effects in the plan library must be reversed
- Plans added for obligations can be removed
- Plans suppressed for prohibitions must be restored

Norm Outcomes

Deontic Modality	Activation Condition	Expiration Condition	Outcome
obligation(O)	True	True	Ignore norm
	True	False	Adopt plan to achieve O (if O is a world state) or adopt plan to execute O (if O is an action)
	False	False	Add plan to achieve O (if O is a world state) or add plan that includes O (if O is an action) to PL when activation holds
	False	True	Ignore norm
prohibition(P)	True	True	Ignore norm
	True	False	Drop intentions to achieve P and suppress plans that achieve P (if P is a world state) or drop intentions that include P and suppress plans that include P (if P is an action)
	False	False	Add plan to suppress plans that achieve P (if P is a world state) or plans that include P (if P is an action) when activation holds
	False	True	Ignore norm

NORMATIVE AGENTSPEAK(L)

Meta-reasoning Operators

- Implementation created using *Jason*
- Extended with meta-level actions:

Action	Effect
.plan_steps(P,S)	takes a plan P and unifies its plan steps as a list of literals with S
.plan_conseq(P,C)	takes a plan P and unifies its declarative consequences with C
.action(A)	succeeds if A refers to an action
.literal(L)	succeeds if L to a literal
.remove_plan(P)	removes P from the plan library
.suppress_plan(P)	suppresses the specified plan (prevents from being executed)
.unsuppress_plan(P)	allows a previously suppressed plan to be executed

Plan Modification Strategies

```
@prohibitionStart(Prohibition)
+!Start : true
 <- !findPlansWithAction(Prohibition, SPlans);
    !suppressPlans(SPlans);
    +suppressedPlans(Prohibition,SPlans).

@prohibitionEnd(Prohibition)
+!End : suppressedPlans(Prohibition,SPlans)
 <- !unsuppressPlans(SPlans);
    .remove_plan(prohibitionStart(Prohibition));
    .remove_plan(prohibitionEnd(Prohibition)).
```

CONCLUSIONS AND FUTURE WORK

Conclusions

- A practical framework for normative processing at the agent level
- Generic enough for application to most modern agent languages
- Working prototype in AgentSpeak(L)
  - www.meneguzzi.eu/felipe/software.html

Future Work

- Current framework is very coarse
- Suppression cannot refer to specific instances of actions/world states
- Future work will allow specific restrictions to be added to norms

Norm-based behaviour modification in BDI agents

Felipe Meneguzzi

Michael Luck

felipe.meneguzzi@kcl.ac.uk