Automation of Normative Reasoning – Rule-based Systems and Answer Set Programming

STUDENT: MARGARITA CHIKOBAVA

SUPERVISOR: PROFESSOR DR. CHRISTOPH BENZMÜLLER

Overview

Motivation: multi-agent systems

Rule Based Systems

Answer Set Programming Systems

Conclusion

Tasks

SIMULATE CASE OF EMERGENCY

TRUST NEGOTIATION

PRICE NEGOTIATIONS

SCHEDULING DECISIONS

Why not just logic

Norms:

- It is forbidden to cross red lights
- Paying a fine is way of “fixing” it

- Logic – look for a conflict
- Is there a conflict?

Norms representation:

deontic logic, other variations of modal logic, and first-order logic

rules: collections of condition/action pairs

sequences of ones and zeros

strategies

Norms representation:

deontic logic, other variations of modal logic, and first-order logic

rules: collections of condition/action pairs

sequences of ones and zeros

strategies

Rule Based System

Norm Negotiation by Guido

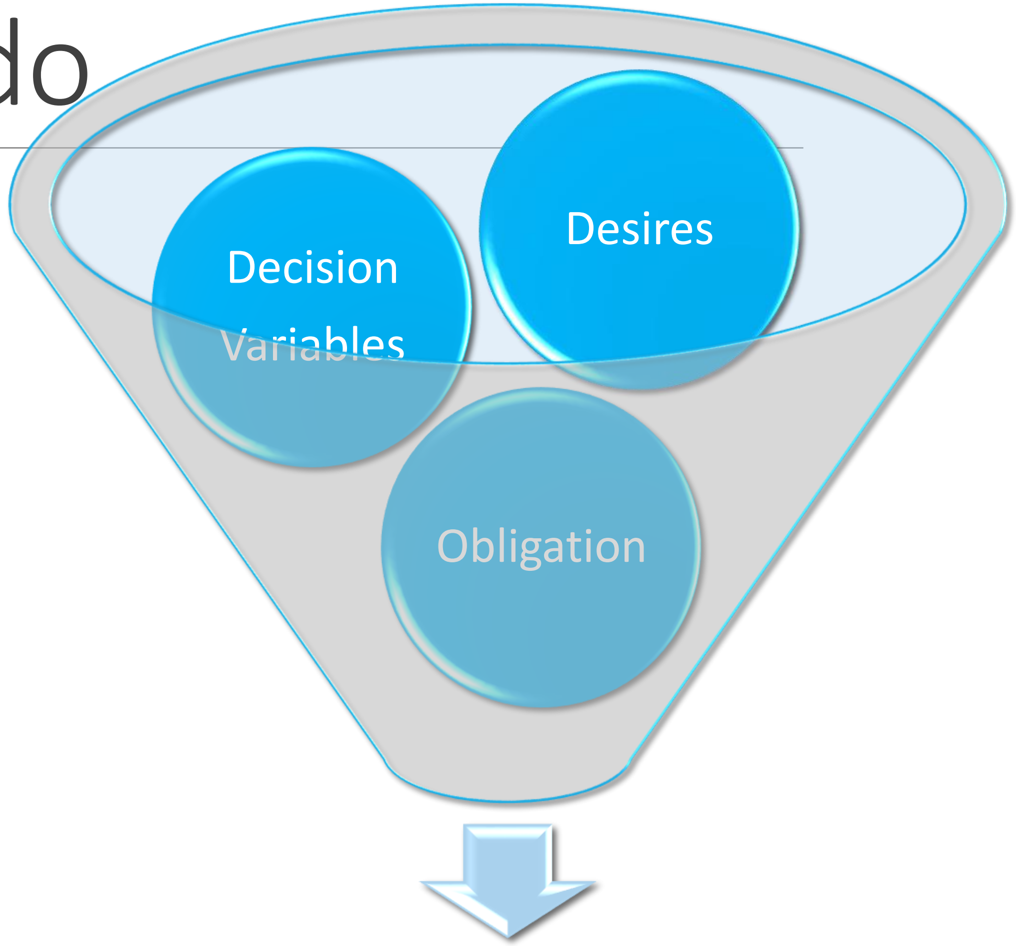
Rule Responder by Paschke and Boley

Norm Negotiation by Guido

Agent:
- Decision Variables
- Desires
- Priority relation

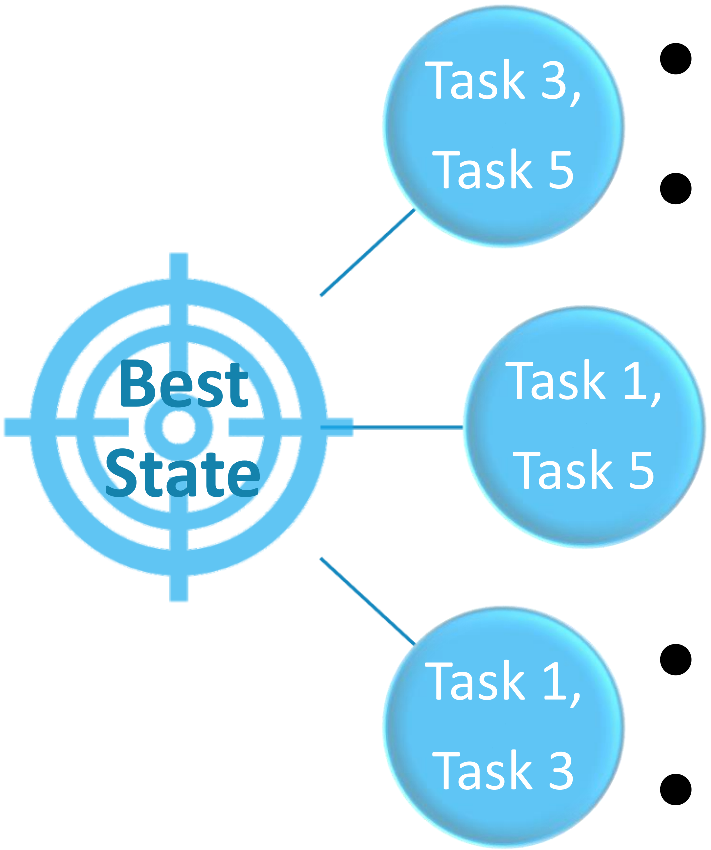
Obligations:
- Obligatory
- Sanction
- Reward

Negotiation Protocol:
- Negotiation Actions
- Agent proposes a deal
- Others accept or not -> success or failure



Social Goal or Norm

Example. Social Goal Negotiation

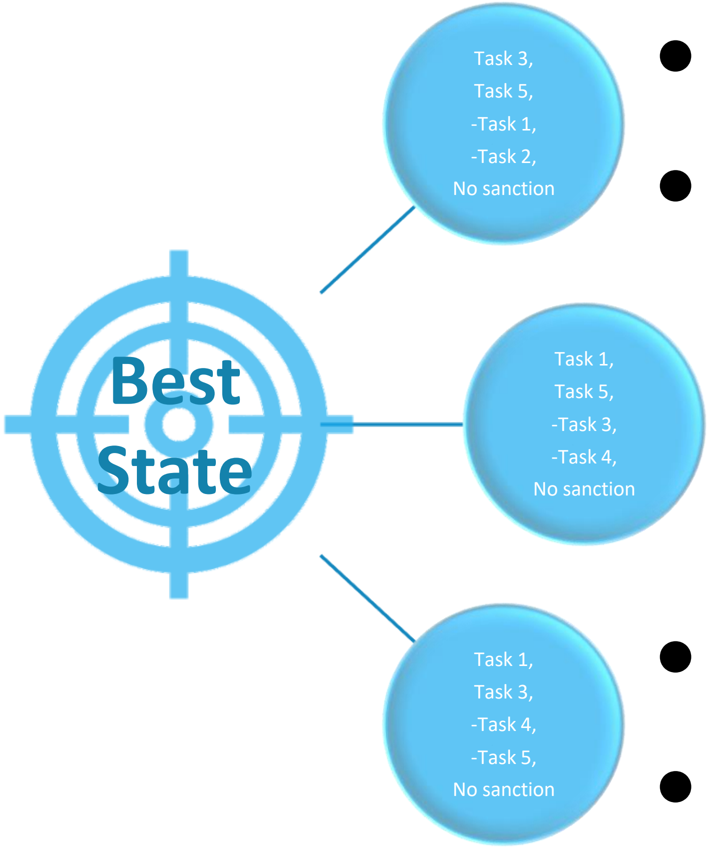


- Task 1
- Task 2

- Task 2
- Task 3

- Task 4
- Task 5

Example . Social Norm Negotiation



- Task 1
- Task 2

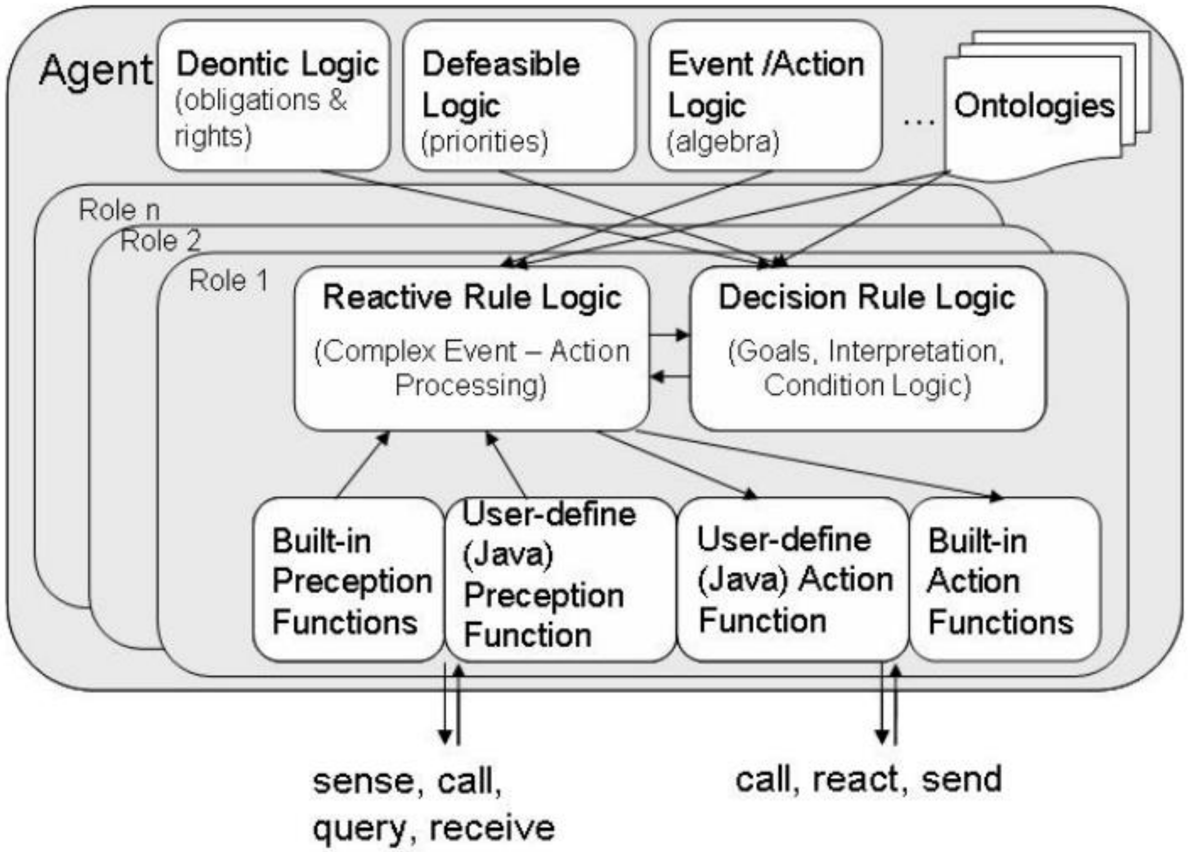
- Task 2
- Task 3

- Task 4
- Task 5

Example

$$
\begin{aligned}
&action_1 : propose(a_1, d_1 = \langle \tau_\delta, \langle s_1, s_2, s_3 \rangle \rangle) \text{ where} \\
&\quad \tau_\delta = \{task_1(a_1), task_2(a_2), task_3(a_3), task_4(a_3), task_5(a_3)\} \\
&action_2 : accept(a_2, d_1) \\
&action_3 : reject(a_3, d_1) \\
&action_4 : propose(a_2, d_2 = \langle \tau'_\delta, \langle s_1, s_2, s_3 \rangle \rangle) \text{ where} \\
&\quad \tau'_\delta = \{task_1(a_1), task_2(a_2), task_3(a_2), task_4(a_3), task_5(a_3)\} \\
&action_5 : accept(a_3, d_2) \\
&action_6 : accept(a_1, d_2)
\end{aligned}
$$

Rule Responder by Paschke and Boley



Example

```
<Rule style="active|messaging|reasoning">

    <meta>   <!-- (semantic) metadata of the rule -->                 </meta>
    <evaluation>   <!-- intended  semantics  -->                 </evaluation>
    <qualification> <!-- e.g. qualifying rule declarations, e.g.
                        priorities, validity, strategy -->   </qualification>
    <quantification> <!-- quantifying rule declarations     </quantification>
    <scope>  <!-- scope of the rule e.g. a local rule module -->     </scope>
    <oid>     <!-- object id of the rule -->                           </oid>

    <on>      <!-- event part -->                                      </on>
    <if>      <!-- condition part -->                                  </if>
    <then>    <!-- (logical) conclusion part -->                     </then>
    <do>      <!--  action part -->                                    </do>
    <after>   <!-- postcondition part after action,
                   e.g. to check effects of execution -->           </after>
    <else>    <!-- (logical) else conclusion -->                     </else>
    <elsedo>  <!-- alternative/else action,
                    e.g. for default handling -->                   </elsedo>
</Rule>
```

Answer Set Programming

Institutions instead of agents

institutional fluent (IF) instead of obligations:

- Institutional Power
- Permission
- Obligation

Institutional state – set of IF

institutional states change due to events

Example

$$\mathcal{E}_{obs} = \{\texttt{shoot}, \texttt{startwar}, \texttt{declaretruce}, \texttt{callup}, \texttt{provoke}\} \quad (1)$$

$$\mathcal{E}_{instact} = \{\texttt{conscript}, \texttt{murder}\} \quad (2)$$

$$\mathcal{E}_{viol} = \{\text{viol}(\texttt{shoot}), \text{viol}(\texttt{startwar}), \text{viol}(\texttt{declaretruce}), \text{viol}(\texttt{callup}), \text{viol}(\texttt{provoke}), \text{viol}(\texttt{conscript}), \text{viol}(\texttt{murder})\} \quad (3)$$

$$\mathcal{D} = \{\texttt{atwar}\} \quad (4)$$

$$\mathcal{W} = \{\text{pow}(\texttt{conscript}), \text{pow}(\texttt{murder})\} \quad (5)$$

$$\mathcal{M} = \{\text{perm}(\texttt{shoot}), \text{perm}(\texttt{startwar}), \text{perm}(\texttt{declaretruce}), \text{perm}(\texttt{callup}), \text{perm}(\texttt{provoke}), \text{perm}(\texttt{conscript}), \text{perm}(\texttt{murder})\} \quad (6)$$

$$\mathcal{O} = \{\text{obl}(\texttt{startwar}, \texttt{shoot}, \texttt{murder})\} \quad (7)$$

$$\begin{aligned}
\mathcal{C}^{\uparrow}(\mathcal{X},\mathcal{E}) : \quad & \langle\{\neg atwar\}, \texttt{startwar}\rangle \rightarrow \{atwar\} && (8)\\
& \langle\{\neg atwar\}, \texttt{provoke}\rangle \rightarrow \{\text{obl}(\texttt{startwar}, \texttt{shoot}, \texttt{murder})\} && (9)\\
& \langle\emptyset, \texttt{conscript}\rangle \rightarrow \{\text{perm}(shoot)\} && (10)\\
& \langle\emptyset, \texttt{startwar}\rangle \rightarrow \{\text{pow}(conscript)\} && (11)\\
\mathcal{C}^{\downarrow}(\mathcal{X},\mathcal{E}) : \quad & \langle\{atwar\}, \texttt{declaretruce}\rangle \rightarrow \{atwar\} && (12)\\
& \langle\emptyset, \texttt{declaretruce}\rangle \rightarrow \{\text{perm}(shoot)\} && (13)\\
& \langle\emptyset, \texttt{declaretruce}\rangle \rightarrow \{\text{pow}(conscript)\} && (14)\\
\mathcal{G}(\mathcal{X},\mathcal{E}) : \quad & \langle\emptyset, \texttt{callup}\rangle \rightarrow \{conscript\} && (15)\\
& \langle\emptyset, \texttt{viol}(\texttt{shoot})\rangle \rightarrow \{murder\} && (16)
\end{aligned}$$

$$S_0 = \{\text{perm}(callup), \text{perm}(startwar), \text{perm}(conscript), \text{perm}(provoke), \text{pow}(murder), \text{perm}(murder)\} \quad (17)$$

Conclusion

Normative multi-agent systems have broad application domain

Norm depends on domain

Norm representation depends on norm

Approach for automation depends on norm representation

Thank you for your attention!

Questions?