Seminar: Normative Reasoning and Machine Ethics

Dozent: Prof. Dr. habil Christoph Benzmüller

Wintersemester 2019/2020

Freie Universität Berlin

Ausarbeitung

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

Student:

Margarita Chikobava

Informatik (Master)

Modeling social problems and simulating complex social conditions and interactions using traditional mathematical models is not easy. Ideas from the various branches of the social sciences have been explored to solve these problems in computer science. Multiagent systems were developed as a result of such collaboration. The agents in such systems may be subject to mechanisms for adjusting their behavior at some level in order to orchestrate a global behavior to achieve a global goal (Fagundes, 2012).

Traditionally normative systems have been studied in philosophy, sociology, law, and ethics, and during the past thirty years they have been studied in deontic logic in computer science. The research focus changes from logical relations among norms, to, for example, agent decision making, and to systems in which norms are created and in which agents can play the role of legislators.

There are several conferences and workshops in this area. For example, the eighth conference on Deontic Logic in Computer Science in 2006 in Utrecht, the Netherlands had as special focus "artificial normative systems", and the seventh conference in 2004 in Madeira, Portugal had as special theme "deontic logic and multiagent systems." Third workshop on normative multiagent systems was co-located in Luxembourg in July 2008 and had as special topic "security and trust" (Boella G. a., Introduction to the special issue on normative multiagent systems, 2008). The domain of usage of such systems that already help to simulate human behavior, for instance in emergency situations is very broad.

In (Pan, 2005) authors explain that to calculate the number, width and distribution of exits it is assumed that occupants would evenly utilize all available exits to escape in case of an emergency. In emergency situations people tend to exit a building following the exits that they are familiar with, the one closest to their car or the one that they think to be the safest one. Besides, in crisis people don't always act rational. As a result, egress designs often fail to meet their functional expectations when a real emergency occurs. So, authors use a *multi-agent system to simulate human behavior* in emergency situations for better egress designs.

Another system handles trust on the internet. It was developed for access control and is based on digital credentials that encode properties (such as subscription ownership, membership to certain organizations, date of birth, etc.). With this system, users can formulate policies to control which pieces of sensitive information that may be encoded in their credentials they are comfortable to disclose. (Bonatti, 2010)

Multi-agent systems are also used in e-commerce. Some of the systems aim to use autonomous agents to completely substitute for humans to automate basic e-commerce

activities such as: product brokering, merchant brokering, negotiations, payment etc. (Badica, 2006)

Other application domain is support for decision making, for instance in scheduling, where the rule-based system predicts which activities are conducted where, when, for how long, with whom and which transport mode is involved. (Janssens, 2004)

Regardless of the specific domain multi-agent system are controlled by some set of rules to regulate their global behavior in order to achieve some goals. This is very close to the definition of norm as it is used in social sciences: *norms* are treated as behaviors which ought to be displayed by members of a group when in a given context. (Boella G. a., 2006) This means that ideas and concepts realized in the multi-agent systems can be translated into *normative multi-agent systems*, in which agents can decide whether to follow the explicitly represented norms, and the normative systems specify how and in which extent the agents can modify the norms.

In order to implement a normative multi-agent system several issues should be addressed. *Agents* can *join a system* whose norms are not known, which means that *norms* should be *communicated to a new agent*. Norms should also be *distributed among other agents*, for example, when new norms emerge. In guaranties that the agent can find a new coalition to achieve its goals. *Norm violations* and norm compliance should be detected as well. Some of these functions can be performed by agents themselves, other should be delegated to the multiagent infrastructure. (Boella G. a., 2008)

To address all these issues in implementation of a multi-agent systems, norms must be specified in a way that enables them to be processed by artificial agents. In literature they differentiate at least four kinds of representation of norms in multi-agent systems: formal logic, rule-based, binary strings and represented as strategies in game theory.

Formal logic includes modal logic and deontic logic. Modal logic is an extension of classical formal logic that can handle the necessary and the possible. Deontic logic expands modal logic and can handle obligations, permissions, and prohibitions. Deontic logic is broadly used in the normative systems from legal theory and is a popular representation scheme within the normative agent community. In the existing literature, deontic logic, other variations of modal logic, and first-order logic, have been used to develop normative agent architectures, extend existing agent models, and specify illegal behavior and its consequences. (Hollander, 2011)

In rule-based systems, norms are represented as collections of condition/action pairs. To handle norms an inference engine and a working memory are used. In normative systems, it is

common to find that the condition/action pairs encode normative behaviors and their associated contexts. This representation format is commonly used by systems that take advantage of offline design, where the norms are coded directly into the agent's decision making. (Hollander, 2011)

Binary strings represent sequences of ones and zeros, where each digit corresponds to the presence (in the case of a one) or absence (in the case of a zero) of a norm. This representation enables dealing with norms on an abstract level, which is often used in research to examine the transmission or emergence of norms in a population. (Hollander, 2011)

In normative multi-agent systems based on game theory, each agent can make its own decisions: a choice that yields a corresponding payoff. At each round of the game, the agents attempt to maximize their payoff. They choose an action based on what they await their opponent to choose. Norms are represented by the strategies that an agent uses to make these decisions. If a considerable number of agents in the population plays by the same strategy a norm emerges. Like the condition/action pairs of rule-based systems, this scheme is commonly used in systems where the norms are designed and encoded offline. (Hollander, 2011)

We will consider systems working with these two norm representations: rules and sequences of ones and zeros, starting with rules and norm negotiation proposed in (Boella G. a., 2007). The proposed framework is based on rule-based systems and can be used both for social goal negotiation and norm negotiation. An agent is described by a set of boolean variables, including decision variables, which describe decisions this agent can perform, and desires used to decision making. Desire rules can be conflicting, and the way the agent resolves its conflicts is described by a priority relation that expresses its agent characteristics.

Every agent has obligations, which can be sanction- and reward-based. Violations and sanctions are the consequences of not fulfilling a norm. Agents negotiate with each other to decide on common goal and common norm. One agent can propose the deal, other could accept it or not, which will lead to success or failure.

During social goal negotiation, a negotiation protocol is used, which can be described by a sequence of negotiation actions which either lead to success or failure. When an agent has made such a proposal, then the other agents can either accept or reject it. Moreover, they can also end the negotiation process without any result. Let’s consider a system with

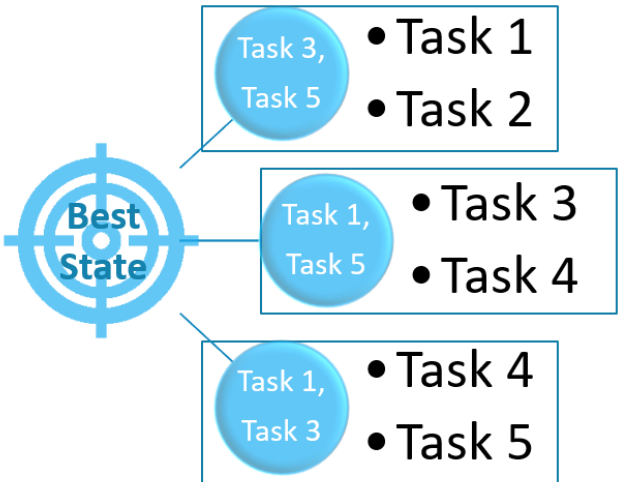

Figure 1 an example of multi-agent system with three agents

three agents depicted in Figure . Every agent has a set of actions that it can perform (bullet point) and a set of goals (in the blue circle). To investigate the common goal, they combine their goals into one common goal with a disjunction. In this example the common goal would be {Task 1, Task 2, Task 3, Task 4, Task 5}.

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

Figure 2 norm negotiation protocol

It is not that simple with social norm negotiation. Each agent desires not to perform any tasks. These desires are weaker (have lower priority) than the desire that the other tasks are performed. Moreover, each agent desires not to be sanctioned. This leads to the more complicated norm communication protocols. An example can be found in Figure , where agent 1 proposes a solution, where task 1 is performed by agent 1, task 2 is performed by agent 2 and agent 3 should perform tasks 3, 4 and 5. This deal is accepted by agent 2 but rejected by agent 3, because it should perform too much tasks. As a response on the rejection by agent 3, agent 2 proposes another deal and proposes to perform task 3 instead of agent 3. This deal is accepted by agent 1 and agent 3.

Another example of rule-based system is presented in (Paschke, 2007). Rule Responder is an open source framework, which helps to create multi-agent systems for virtual organizations. They support collaborative rule-based agent networks on the Semantic Web, where independent agents exchange event messages and cooperate to achieve (collaborative) goals. Rule Responder agents communicate using different negotiation protocols which allows implementing different agent coordination. Messaging reaction rules Reaction RuleML are used for communication between the distributed agent inference services to semantically annotate primitives, such as speech acts, or message content information, such as deontic organizational norms, purposes or goals and values etc. All these semantically annotated primitives are represented as ontological concepts, using RDF Schema or OWL.

The (semi-)autonomous agents use rule engines and semantic eb rules. These semantic web rules describe and execute derivation and reaction logic. They declaratively implement the organizational semiotics and the different distributed system/agent topologies with their

negotiation/coordination mechanisms. Rule Responder support several reasoning engines among others, Prova with a Prolog-like language, OO jDREW with RuleML/POSL, and DR-Device with a defeasible logic rule language. The infrastructure of Rule Responder is flexible and can build normative organizations and implement social and organizational concepts without restrictions from modelling/programming language. Rule Responder permits agents to use local platform specific languages and engines. To communicate with each other despite own implementation language agents should used messages annotated with RDF/OWL.

Another type of multiagent-based system is based on norm representation as sequences of ones and zeros and is used in the INSTAL framework represented in (Cliffe, 2006). The norm reasoning is implemented using answer set programming techniques. The INSTAL framework is a normative framework, which is used to specify, verify and reason about the norms used in an open distributed system. It models an institution as a set of institutional states that change over time due to events introduced to the system. Every institution is described by a set of institutional fluents that true or false at every timestep. Fluents are classified into domain fluents and normative fluents. The former ones depend on the institution being modelled, for example “A owns something”. The later ones are common to all specifications. There are three types of normative fluents: institutional power, permission and obligation.

Institutional power represents the institutional capability to introduce an event to the system and make some changes in the institutional state. Without institutional power, the event may not be brought about and has no effect. For example, a marriage ceremony will only bring about the married state, if the person performing the ceremony is empowered so to do.

Permission fluents represent the property that some event may occur without causing a violation. If an event occurs, and that event is not permitted, then a violation event is generated.

Obligation fluents are modelled as the counterpart of permission fluent. It states that a event is obliged to occur before a given deadline event. An example of implementation of the deadline event is a timeout. It is associated with a specified violation. If an obligation fluent holds and the obliged event occurs, then the obligation is said to be satisfied. If the corresponding deadline event occurs, then the obligation is said to be violated.

The authors illustrate their system with an example describing two countries on the brink of the war. There is an agreement between these countries. If they are at war, it is permitted to shout the citizens of the of the country, otherwise it is not. If this agreement is violated, the country is obligated to start the war.

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

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

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

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

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

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

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

$$\mathcal{C}^{\uparrow}(\mathcal{X}, \mathcal{E}): \langle\{\neg atwar\}, \texttt{startwar}\rangle \rightarrow \{atwar\} \quad (8)$$

$$\langle\{\neg atwar\}, \texttt{provoke}\rangle \rightarrow \{\text{obl}(\texttt{startwar}, \texttt{shoot}, \texttt{murder})\} \quad (9)$$

$$\langle\emptyset, \texttt{conscript}\rangle \rightarrow \{\text{perm}(shoot)\} \quad (10)$$

$$\langle\emptyset, \texttt{startwar}\rangle \rightarrow \{\text{pow}(conscript)\} \quad (11)$$

$$\mathcal{C}^{\downarrow}(\mathcal{X}, \mathcal{E}): \langle\{atwar\}, \texttt{declaretruce}\rangle \rightarrow \{atwar\} \quad (12)$$

$$\langle\emptyset, \texttt{declaretruce}\rangle \rightarrow \{\text{perm}(shoot)\} \quad (13)$$

$$\langle\emptyset, \texttt{declaretruce}\rangle \rightarrow \{\text{pow}(conscript)\} \quad (14)$$

$$\mathcal{G}(\mathcal{X}, \mathcal{E}): \langle\emptyset, \texttt{callup}\rangle \rightarrow \{conscript\} \quad (15)$$

$$\langle\emptyset, \texttt{viol}(\texttt{shoot})\rangle \rightarrow \{murder\} \quad (16)$$

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

Figure 1institutional model

The institutional model is depicted in Figure 3 and shows that a country can observe a shouting, declare the war or declare a truce (1). Two other events can be introduced to the systems by citizens: join military forces or murder someone (2). In (3) we can see which violations are covered by the system. In (4) is stated only one domain fluent – country can be at war. Institutional fluent, namely empowerments, permissions and obligation, are depicted in (5-7). If a country decides to start a war in time of peace, it results in the institutional state changing to war (8). In (9) the obligation for each country (to start a war first before shooting to avoid committing a murder whenever being provoked during a period of peace) is depicted. After conscription it can shoot (10) and conscription also is followed by the start of the war (11). The war can be ended by declaring the truce (12), which also revokes the permission to shoot (13) and power to conscript (14). After the callup command issued by a country, the institution will generate conscription when empowered (15). In case of shooting violation, the

institution will raise the murder event (16). In (17) we see the initial state of the system, which can be changed by introducing new events to the system.

All three depicted systems are normative multi-agent systems. Normative multi-agent systems have broad application domain. Norms, which control the behavior of the agents in the system, depends on domain Norm representation should be chosen based on norm itself. After that decision, the approach for automation corresponding to norm the representation can be chosen and implemented.

Bibliography

Badica, C. a. (2006). Implementing rule-based mechanisms for agent-based price negotiations. *Proceedings of the 2006 ACM symposium on applied computing*, (pp. 96-100).

Boella, G. a. (2006). Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 71-79.

Boella, G. a. (2007). Norm negotiation in multiagent systems. *International Journal of Cooperative Information Systems*, 97-122.

Boella, G. a. (2008). Introduction to the special issue on normative multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 1-10.

Bonatti, P. a. (2010). A Rule-based Trust Negotiation System. *IEEE Transactions on Knowledge and Data Engineering*, 1507-1520.

Cliffe, O. a. (2006). Answer Set Programming for Representing and Reasoning About Virtual Institutions., (pp. International Workshop on Computational Logic in Multi-Agent Systems).

Fagundes, M. S. (2012). Using normative markov decision processes for evaluating electronic contracts. *AI Communications*, 1-17.

Hollander, C. D. (2011). The current state of normative agent-based systems. *Journal of Artificial Societies and Social Simulation*.

Janssens, D. a. (2004). Improving performance of multiagent rule-based model for activity pattern decisions with Bayesian networks. *Transportation research record*, 75-83.

Pan, X. a. (2005). A multi-agent based simulation framework for the study of human and social behavior in egress analysis. *Computing in civil engineering*, 1-12.

Paschke, A. a. (2007). Rule responder: RuleML-based agents for distributed collaboration on the pragmatic web. *Proceedings of the 2nd international conference on Pragmatic web*, (pp. 17-28).