Neural Symbolic Architecture for Normative Agents

(Extended Abstract)

Guido Boella
University of Torino
guido@di.unito.it

Silvano Colombo Tosatto
University of Luxembourg
colombotosatto.silvano@gmail.com

Artur d'Avila Garcez
City University London
aag@soi.city.ac.uk

Valerio Genovese
University of Luxembourg
valerio.genovese@uni.lu

Dino Ienco
University of Torino
ienco@di.unito.it

Leendert van der Torre
University of Luxembourg
leon.vandertorre@uni.lu

ABSTRACT

In this paper we propose a neural-symbolic architecture to represent and reason with norms in multi-agent systems. On the one hand, the architecture contains a symbolic knowledge base to represent norms and on the other hand it contains a neural network to reason with norms. The interaction between the symbolic knowledge and the neural network is used to learn norms. We describe how to handle normative reasoning issues like contrary to duties, dilemmas and exceptions by using a priority-based ordering between the norms in a neural-symbolic architecture.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Applications and Expert Systems

General Terms

Algorithms, Theory, Legal Aspects

Keywords

Norms, Computational architectures for learning, Emergent behavior, Logic-based approaches and methods

1. NEURAL-SYMBOLIC ARCHITECTURE

Figure 1 visualizes the architecture of an agent adopting a *neural-symbolic system* [2]. The agent builds a network from the symbolic knowledge it possesses. The neural network is used to process the data incoming from the surrounding environment. The output resulting from the neural network are the actions the agent has to perform. Furthermore the network can be trained by feeding it with instances representing the correct behaviors in certain situations that the agent cannot perform due to its incomplete knowledge. After the training, the resulting neural network can be used to improve the symbolic knowledge of the agent as explained in details in [2]. The improved knowledge base can be used to build a new neural network that the agent will use to interact with the environment. The new neural network is an improvement over the old one due to the new knowledge added within the existing symbolic knowledge after the training. The normative agent is capable to automatically improve its performance by interacting with the surrounding environment.

Cite as: Neural Symbolic Architecture for Normative Agents (Extended Abstract), G. Boella, S. Colombo Tosatto, A. d'Avila Garcez, V. Genovese, D. Ienco and L. van der Torre, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. XXX-XXX.

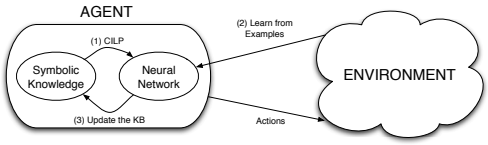

Figure 1: Normative Agent representation.

2. NORMATIVE AGENT

The normative agent has to reason with norms. To do so we use I/O logic [3] to represent the norms contained in the knowledge base representing the symbolic knowledge. I/O logic rules (α, β). Both α and β represent a set of literals in conjunction. β represents the input, the antecedent of the rule and determines whenever β is observed the activation of the rule. Instead α represents the output, the consequent of the rule which is the obligation or permission returned in the result whenever the rule is activated.

In order to allow the agent to efficiently reason about norms, we have to handle some of the issues known in normative reasoning, like contrary to duties, dilemmas and exceptions. We are going to use a priority-based ordering in some of these problems in order to handle them [1].

Priority-based ordering: By introducing a priority-based ordering between the rules, we are able to decide when two different rules were activated at the same time which one has to be and the one which should not. by enforcing a priority-based ordering between two rules, in the case where both are activated, then only the one with the higher priority is. We use the negation as failure to embed the priority concept within the rules. We are going to explain it with an example, we need to consider two rules: $r_1 = a \wedge b \rightarrow O(c)$ and $r_2 = a \wedge d \rightarrow O(e)$ and having a priority-based ordering $r_1 \succ r_2$ which means that the first rule has the priority over the second. We embed the priority within the rule which is overcome because is the one that must be suppressed by the activation of the other. To embed the priority we modify the antecedent of the rule with the lower priority in a way that it is not activated if the other is. To obtain so we add to the antecedent of this rule the negation as failure of the literals which are only included in the antecedent of the rule with the higher priority. By applying this process to our example we obtain a new modified rule: $r_2' = a \wedge d \wedge \sim b \rightarrow O(e)$ which is not activated if b is observed, because otherwise also r_1 would be.

Contrary to duty: A contrary to duty is composed by two rules, one is used to regulate the optimal situation and the other has to be

applied in a sub-optimal situation where the first cannot. A classic example due to Sergot [4] refers to a situation where a cottage should not have a fence, the rule $r_1 : \top \rightarrow O(\neg f)$ describes the ideal situation. Instead the rule $r_2 : f \rightarrow O(w)$ can be used to handle a sub-optimal situation where a cottage has a fence. The rule states that if the cottage has a fence it should at least be white. The problem with contrary to duties can be noted when considering the sub-optimal situation. In the Sergot's example the sub-optimal situation refers to the case where the cottage has a fence f. If we apply the rules to the sub-optimal situation we obtain two obligations: $\neg f$ and w, the obligation to not have a fence is unfulfillable because the cottage already has it. We want to avoid to produce obligation that cannot be achieved. By setting a priority-based ordering between the rules $r_2 \succ r_1$ we state that we do not want to apply the first rule whenever the second holds. This because, if the second rule can be applied, means that we are in a sub-optimal situation where the first rule consequent cannot be complied.

Dilemma: A dilemma is a controversial situation that can occur in normative reasoning. It happens when analyzing a situation, two different rules produces contradicting obligations that have to be fulfilled. A classic example of dilemma is Sartre's soldier, it can be described with two rules, the first says that everyone should not kill $\top \rightarrow \neg O(k)$ and the second states that a soldier has the duty to kill his enemies $s \rightarrow O(k)$. The dilemma is generated when both rules are applied in the same circumstance. If we consider the case of an ordinary person (which is not a soldier) then only the first rule is applied returning the obligation $\neg k$ which does not produce a dilemma. Instead if we consider the case where a soldier is involved, both rules are applied and the outputs produced are both $\neg k$ and k which is a moral dilemma that the soldier has to cope with. Having described the structure of a dilemma problem, we have decided not to use a priority-based ordering between the rules to enforce a decision. Instead, by considering that dilemmas are part of everyday life decisions, we decided to leave to leave the dilemma open for the agent which will have to make a decision considering that both choices are suitable.

Exception: An exception refers to a situation where a rule should be applied instead of another one. We can consider a clarifying example, a general rule is that a person should not activate the fire alarm $r_1 : \top \rightarrow O(\neg a)$ but in the case where someone spots a fire, then he should activate the alarm $r_2 : f \rightarrow O(a)$. If we consider the two rules and a situation where someone spots a fire, then both rules are activated producing the dilemma a and $\neg a$ which is undesirable, because we want that when someone spots a fire he must activate the alarm. To address this problem we use a priority-based ordering between the rules $r_2 \succ r_1$. In this way by activating the second rule, it inhibits the first one resulting in the single obligation a to trigger the fire alarm.

Permissions: In normative reasoning the permission is another important element, because in some scenarios it is important to define also when it is permitted to do something. We can suppose that the symbolic knowledge of the agents contains both rules that produce obligations and rules that have permissions as consequents. In our case we are going to consider that something is permitted if not explicitly forbidden, so we do not explicitly represent permission in the neural network translation. We instead use rules that produce permissions to undercut obligation rules with which they are in conflict. To do so we use a priority-based ordering between the rules. Considering two generic rules $r_1 : a \rightarrow O(\neg c)$ and $r_2 : b \rightarrow P(c)$, we can see that the permission in the consequent of the second rule is in contradiction with the obligation of the first. By applying a priority-based ordering $r_2 \succ r_1$ we use the rule with the permission to inhibit the first. After applying the translation on r_1 due to the priority-based ordering, r_2 will not be translated into the network.

3. NORMATIVE NETWORK

The neural network is built from the symbolic knowledge, in this way the resulting network is already capable to analyze some situations and returning for those the correct behaviors without any training. Due to using a symbolic knowledge containing normative rules expressed with I/O logic [3], we have to use a variant of the CILP translation algorithm already described in [2]. We keep the inputs and the outputs of the neural network well separated as for I/O logic, because we do not use feedback connections in the network. This means that the outputs produced by the network are not reused as input and fed again to the network. In this way we do not need to wait for the network to stabilize but with a single step it is sufficient to obtain the outputs for the situation that is being analyzed. Output neurons are interpreted as obligations when positive and as denials when the label of the output is a classically negated atom.

Normative-CILP is a (sound) algorithm to embed I/O rules into a feedforward NN.

N-CILP

1. For each literal α_{i_j} $(1 \leq j \leq m)$ in the input of the rule. If there is no input neuron labeled α_{i_j} in the input level, then add a neuron labeled α_{i_j} in the input layer.
2. Add a neuron labeled N_k in the hidden layer.
3. If there is no neuron labeled β_{o_1} in the output level, then add a neuron labeled β_{o_1} in the output layer.
4. For each literal α_{i_j} $(1 \leq j \leq n)$; connect the respective input neuron with the neuron labeled N_k in the hidden layer with a positive weighted arc.
5. For each literal $\sim \alpha_{i_h}$ $(n+1 \leq j \leq m)$; connect the respective input neuron with the neuron labeled N_k in the hidden layer with a negative weighted arc[1].
6. Connect the neuron labeled N_i with the neuron in the output level labeled β_{o_1} with a positive weighted arc[2]

The N-CILP has been implemented and tested over a case study based on RoboCup rules. A java implementation of N-CILP is available at

`http://www.di.unito.it/~genovese/tools`.

Acknowledgements: Valerio Genovese and Silvano Colombo Tosatto are supported by the National Research Fund, Luxembourg.

4. REFERENCES

[1] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. cognitive science quarterly. In *Cognitive Science Quarterly*, volume 2(3-4), pages 428–447, 2002.

[2] A. d'Avila Garcez, K. Broda, and D. Gabbay. *Neural-Symbolic Learning Systems*. Springer, 2002.

[3] D. Makinson and L. van der Torre. Input-output logics. *Journal of Philosophical Logic*, 29, 2000.

[4] H. Prakken and M. J. Sergot. Dyadic deontic logic and contrary-to-duty obligations, 1997.

[1]The connections between these input neurons and the hidden neuron of the rule represents the priorities translated with the *NAF*.

[2]Each output in the rules is considered as a positive atom during the translation, this means that if we have a rule with a negative output $\neg\beta$, in the network we translate an output neuron labeled β' that has the same meaning of $\neg\beta$ but for the translation purpose can be treated as a positive output.