Embedding Normative Reasoning into Neural Symbolic Systems

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

Silvano Colombo Tosatto
University of Luxembourg
silvano.colombotosatto@uni.lu

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

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

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

Abstract

Normative systems are dynamic systems because their rules can change over time.

Considering this problem, we propose a neural-symbolic approach to provide agents the instruments to reason about and learn norms in a dynamic environment.

We propose a variant of d'Avila Garcez et al. Connectionist Inductive Learning and Logic Programming(CILP) System to embed Input/Output logic normative rules into a feed-forward neural network. The resulting system called Normative-CILP(N-CILP) shows how neural networks can cope with some of the underpinnings of normative reasoning: *permissions*, *dilemmas*, *exceptions* and *contrary to duty* problems.

We have applied our approach in a simplified RoboCup environment, using the N-CILP simulator that we have developed. In the concluding part of the paper, we provide some of the results obtained in the experiments.

1 Introduction

In artificial social systems, norms and policies are mechanisms to effectively deal with coordination in normative multi-agent systems. An open problem in AI is how to equip agents to deal effectively with norms (and policies) that change over time [Boella *et al.*, 2009], either due to explicit changes by legislators, or due to the interpretation process by those agents who are in charge of applying the law (e.g, judges).

In the work of [Corapi *et al.*, 2010], they focused on refine existing knowledge about the norms by using inductive learning. Differently in game-theoretic approaches [Sen and Airiau, 2007; Boella and van der Torre, 2006; Shoham and Tennenholtz, 1997], few machine learning techniques have been applied to tackle open problems like learning and/or revising new norms in open and dynamic environments.

In this paper we use Input/Output (I/O) logic [Makinson and van der Torre, 2000], a symbolic formalism used to represent and reason about norms. We study how to represent I/O within the computational model of neural networks, in order to take advantage of their ability to learn, by addressing the following research question:

- How to define a formal framework combining I/O logic and neural-symbolic computation for normative reasoning?

Among other formalisms used in normative systems, we choose I/O logic because it presents a strong (and natural) similarity with neural networks: both have a separate specification of inputs and outputs. We exploit such similarity first to encode knowledge expressed in terms of I/O rules into neural networks, and then to use the neural network to reason and learn new norms in a dynamic environment.

Methodologically, we adopt the Neural-Symbolic paradigm of [d'Avila Garcez *et al.*, 2002] which embeds (symbolic) logical programs into feed-forward neural networks. Neural-symbolic systems provide translation algorithms from symbolic logic to neural networks and vice-versa. The network is used for robust learning and computation, while the logic provides (i) background knowledge to help learning (when the logic is translated into the neural network) and (ii) high-level explanations for the network models[1] (when the trained neural network is translated into the logic). A sound translation for the (i) step is done by using the CILP system [d'Avila Garcez *et al.*, 2002].

In normative reasoning there are some problems which have to be handled. These problems are: *permissions*, *dilemmas*, *contrary to duties* and *exceptions*. A normative agent, is an agent capable to behave within an environment regulated by norms, must be able to handle the situations listed above. A way to handle such situations is by using *priorities*. A description about how a normative agent can handle such situations with the use of priorities is described in [Boella *et al.*, 2011].

In particular, we address the following sub-questions:

- How to use priorities with I/O logic rules in order to handle normative reasoning problems?
- How to translate I/O logic into neural networks by using CILP and keeping the soundness of the logic?

We provide a description of the simulator used for testing our approach. The simulator has been written in *Java*[2] and

[1]We are not going to discuss this step in this paper.
[2]www.java.com/

using the package *Joone*[3], a framework to model neural networks.

After describing the simulator we provide the results obtained from some of the experiments made.

The paper is structured as follows. In Section 2 we introduce the neural-symbolic approach, the I/O logic and the architecture of a normative agent. In Section 3 we first describe which restrictions need to be applied to I/O rules. Then how we embed priorities within the rules and at last the role of *permissions* in our approach. In Section 4 we describe the case study used in the experiments. In Section 5 we describe the simulator and the experiments. In Section 6 we present the conclusions.

2 Related work

Neural-Symbolic approach

The main purpose of a *neural-symbolic approach* is to bring together connectionist and symbolic approaches [d'Avila Garcez *et al.*, 2002]. In this way it is possible to exploit the strengths of both approaches and to avoid their drawbacks. With such approach we are able to formally represent the norms governing the normative system. In addition we are also capable to exploit the instance learning capacities of neural networks and their massive parallel computation.

Algorithms like *KBANN*[Towell and Shavlik, 1994] and *CILP*[d'Avila Garcez and Zaverucha, 1999] provide a sound translation of a symbolic representation of the knowledge within a neural network. The advantage of CILP, is that it uses the sigmoid function for its perceptrons. This allows the use of *backpropagation* for learning. In what follows, we use a variant of CILP since we are interested in the integration of reasoning and learning capabilities.

I/O Logic

To describe the norms regulating the system we use I/O Logic [Makinson and van der Torre, 2000]. Rules used in I/O logic are defined in the shape $R_1 = (A, B)$. Both A and B represent sets of literals. The literals contained in A (or in B) can be either in conjunction or disjunction between them. A represent the antecedent of the rule, what must be considered true in order to activate the rule. Instead B is the consequent, what is considered true after the rule has been activated.

I/O logic provides some reasoning mechanisms to produce outputs form the inputs. The first of this mechanisms is the *simple-minded output*. This mechanism does not satisfy the principle of identity. Instead the simple-minded output possess other features like *strengthening input*, *conjoining output* and *weakening output*. The I/O logic also provides other reasoning mechanisms, *basic output*, *reusable output* and *reusable basic output* which allow additional features. Respectively *input disjunction* for the basic output, *reusability* for the reusable output and both for reusable basic output. A detailed description of the I/O logic mechanisms and features can be found in [Makinson and van der Torre, 2000].

In [Boella *et al.*, 2010] it is described how a connectionist approach like neural networks can embed the different features of I/O logic. In this way it is possible by using translation algorithms like KBANN or CILP to reproduce the mechanisms of I/O logic.

[3]http://sourceforge.net/projects/joone/

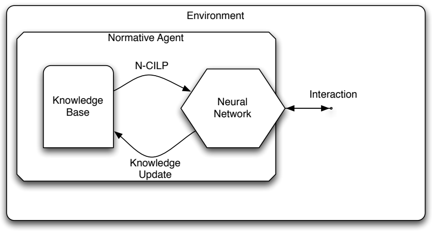

Figure 1: Neural-Symbolic Normative Agent.

Normative agent

An agent is defined as an entity that actively interacts with its surrounding environment and with other agents if we consider a multi-agent system. In this paper we will focus on a single agent, more precisely a *normative agent*. Figure 1 shows a normative agent, an entity that has to act and behave by following the norms regulating the environment where it acts. A more detailed description of what is a normative agent can be found in [Boella *et al.*, 2011].

In this paper we do not focus on which action the agent should execute in a particular situation. For *situation* we mean a particular state of the environment, including all the inputs that the agent can use to make its decisions. Instead we concentrate our efforts into deciding what an agent ought to do and can do while in a particular situation.

The normative agent must be capable to handle the problems that can arise. In normative reasoning some of this problems are *dilemmas exceptions* and *contrary to duties*. Dilemmas occurs when the agent is facing two contradictory obligations. With contradictory obligations we mean two different obligations which cannot be accomplished both. An example is the *Sartre's soldier*, which has the moral obligation to not to kill, but being the soldier he has to fight and kill his enemies. The second problem that an agent may face is the exception. An exception, like the name suggests, occurs during exceptional situations. In these exceptional situations it is possible that a rule which usually has to be applied is overridden by a different one. We can provide an example by considering the rules of a football match. The standard rule is that the players cannot play the ball with their hands. In this case we have an exception if we consider the goalkeeper. This particular player while inside its own goal area, is allowed to use its hands to play the ball. The last problem mentioned is the contrary to duty[Prakken and Sergot, 1996]. In normative reasoning the violation of a rule is not always to be considered a critical failure. In some circumstances is possible to handle the violation by fulfilling alternative obligations. As an example we can consider the situation where we are in a pub with a friend. Supposing that our friend is drinking a beer. The general rule is that we should not spill our friends beer. Considering the unfortunate situation where we accidentally (or not) spill our friend's beer. Our friend has now the possibility to severe our friendship due to our violation. In this situation we still have the possibility to repair to the violation, by considering to buy our friend a new beer.

3 Neural Networks for Norms

3.1 I/O logic for Norms

In order to use I/O logic to represent normative rules, we need to add modalities. We add two different types of modalities, the obligation ($\mathbf{O}$) used to define what the agent is ought to do or prohibited[4] and the permission ($\mathbf{P}$) use to define what is permitted to the agent. We will consider the modalities introduced are unary operators, acting over a single literal. For example $\mathbf{P}(\alpha)$ represents the permission to do α.

Considering again an I/O logic rule: $R_1 = (A, B)$ where A and B are set of literals. By unfolding the set B we can consider all the literals contained in the consequent: $B = \{\beta_1, \beta_2, \ldots, \beta_n\}$. At each literal in the consequent, can be added one of the possible modalities. By doing so what we obtain is a normative rule, a rule which does not states facts but, oughts, prohibitions and permissions for the normative agent. For example: $\mathbf{O}(\beta_1), \mathbf{P}(\beta_2), \mathbf{O}(\beta_3)$, a normative rule with such consequent, would mean that β_1 and β_3 are oughts and β_2 is a permission instead.

3.2 I/O rules restrictions

For the translation we adopt a variant of the CILP algorithm. We use N-CILP, that translates a knowledge base containing I/O logic rules in a neural network.

In order to allow N-CILP to translate I/O logic rules, we have apply some restrictions on the rules. However those restrictions are not crippling the expressivity of the logic.

1. First we need to restrict the antecedent (input) of the rule. We want that the literals in the antecedent are connected by conjunctions only. We see now how this does not harm the expressivity of the logic. Considering a I/O logic rule with a disjunction in the antecedent like the following: $(A_1 \vee A_2, B)$ where A_1 and A_2 are sets of conjuncted literals. For each disjunction we split the antecedent. In this particular case we split the starting rule into two rules with a new antecedent and the same consequent. Obtaining in this case two rules: (A_1, B) and (A_2, B) that considered together allow the same semantics of the starting rule.

2. We restrict the consequent (output) to contain a single literal. If we consider the set of consequents to be constituted by conjuncted literals, then every literal in the set produces a new rule, with itself in the consequent and the same antecedent as the starting rule.

   In this case the logic may lose some expressivity, it may happen if we need disjunctions in the consequent. Disjunctions in the consequent can be used to introduce uncertainty in the system. However due to the fact that we consider normative systems, the rules are used to describe the norms governing the system. We can safely assume that norms are meant to regulate the system and not introduce uncertainty.

3. The last restriction regards the consequent. In addition we have to restrict it to be a positive literal. We address this problem by syntactically considering a negative literal as positive. In example the consequent $\neg\beta_i$ is considered as: β_i'. The newly created literal is semantically still considered negative. Also in this case the logic does not lose expressivity.

3.3 Priorities

Priorities are used to give a partial ordering between rules. This ordering is useful because sometimes between two applicable rules we want to apply only one. This can happen when considering for example *exceptions*.

Here we explain how we encode priorities within the rules by using the *negation as failure* ($\sim$). Considering for example two rules: $R_1 = (A_1 \wedge A_2, \mathbf{O}(\beta_1))$, $R_2 = (A_1 \wedge A_3, \mathbf{O}(\beta_2))$ and a priority relation between them: $R_1 \succ R_2$, where the first rule has the priority. Knowing A_1, A_2 and A_3 are sets of conjuncted literals, we embed the priority into the rule with the lowest priority. To do so we include into the antecedent of the rule with lower priority, the negation as failure of the literals in the antecedent of the higher prioritized rule, that does not appear in the antecedent of the lower priority rule.

Considering for example the two rules given, we have to modify R_2. In this case we need to include in the antecedent of R_2 the part of the antecedent of R_1 that differs, in this case A_2. After embedding the priority within the second rule, it becomes: $R_2' = (A_1 \wedge \sim A_2 \wedge A_3, \mathbf{O}(\beta_2))$.

3.4 Permissions

An important distinction between *oughts* and *permissions*, is that the second ones are not explicitly encoded in the neural network. In our approach we consider that something is permitted to the agent if not explicitly forbidden[5]. Due to this we consider rules with a permission in their consequent to implicitly have the priority over the rules that forbid the same action.

For example considering two rules $R_1 = (A_1, \mathbf{P}(\beta_1))$, $R_2 = (A_2, \mathbf{O}(\neg\beta_1))$. The first rule permits β_1 and the second forbids it. In this case we consider implicitly the following priority relation $R_1 \succ R_2$ to hold.

4 Case study

To test the performance of our approach to normative reasoning we use the RoboCup scenario. For simplicity we focused on the reasoning of a single robot, leaving out the multi-agent aspect of the scenario.

With our approach, the robot does not plan the sequence of the actions. Instead the robot analyzes the current situation and by taking into consideration the rules of the game[Menegatti, 2007], it knows what is ought and what is prohibited.

If we consider that the robot makes its decisions taking into account only the rules, then the robot is acting within a static environment. Because the rules does not change in the middle of the game. In order to add dynamism into the environment

[4]By prohibition we mean the obligation of a negative literal. In example we can have $\mathbf{O}(\neg\chi)$ which means the obligation to do not χ, in other words the prohibition to do χ.

[5]We consider the ought of a negative literal as a prohibition.

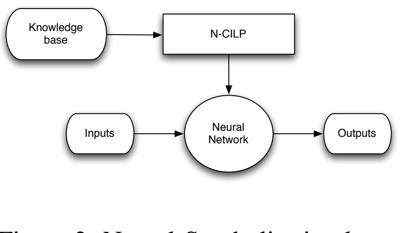

Figure 2: Neural-Symbolic simulator.

we add an additional ruling element. The first ruling element is the referee, which enforces the rules of the game. The additional ruling element is the coach, which demands to the robots to play in a specific way. The coach can introduce new rules or lift some of the existing ones during the game. In this way, a robot that acts in an environment where the coach is involved, sometimes needs to adapt its behavior.

4.1 Knowledge base structure

The knowledge base used by the robot contains both the rules of the game and the coach directions. Both the rules and the directions are shaped in I/O logic rules format. The knowledge base used in the experiments contains 29 rules, including the rules which have a permission in their consequent.

The knowledge base also contains the priority relations between the rules, which are used to resolve possible conflicts among them (like the production of contradicting oughts).

We show some of the rules contained in the knowledge base:

$R_1 : (\top^6, \mathbf{O}(\neg impact\_opponent))$

$R_2 : (\top, \mathbf{O}(\neg use\_hands))$

$R_3 : (goalkeeper \wedge inside\_own\_area, \mathbf{P}(use\_hands))$

$R_4 : (ball \wedge opponent\_approaching, \mathbf{O}(pass))$

The first rule states that a robot should never impact into an opponent. The second rule is again a prohibition, that states that a robot should not use its hands to play the ball. The third rule is different, because states that the goalkeeper is allowed to use its hands while inside its own goal area. The last rule is not from the RoboCup ruling, instead is one of the rules that the coach may have given to robots, to influence their playing behavior. The fourth rule states that if an opponent is approaching the robot handling the ball, then that robot should pass the ball.

5 Simulator and experimental results

5.1 The simulator

In Figure 2 it is shown how the simulator works. The knowledge base contains the the rules that the robot knows. We consider that the priorities are embedded within the rules as described in the previous section. The knowledge base is used as the input for the *N-CILP* translation algorithm.

[6] $\top$ means that the antecedent is always true, in other words the rule is always applied.

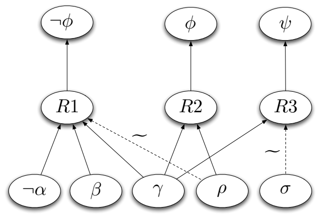

Figure 3: Example of Embedding

N-CILP

Given a knowledge base $\mathcal{KB}$ for each rule $R_k = (\alpha_{i_1} \wedge \ldots \wedge \alpha_{i_n} \wedge \sim \alpha_{i_n+1} \wedge \ldots \wedge \sim \alpha i_m, \mathbf{O}(\beta_{o_1}))$in $\mathcal{KB}$ do:

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 α_{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[7].
6. Connect the neuron labeled N_i with the neuron in the output level labeled β_{o_1} with a positive weighted arc[8]

In [d'Avila Garcez *et al.*, 2002] it is shown how the weights of the resulting neural network can be calculated.

In Figure 3 we show the structure of a neural network constructed with the N-CILP algorithm from the translation of four rules. The rules are $R_1 = (\neg\alpha \wedge \beta \wedge \gamma, \mathbf{O}(\neg\phi))$, $R_2 = (\gamma \wedge \rho, \mathbf{O}(\phi))$, $R_3 = (\gamma, \mathbf{O}(\neg\psi))$ and the *permission rule* $R_4 = (\gamma \wedge \sigma, \mathbf{P}(\psi))$. Between the rules we have a priority ordering $R_2 \succ R_1$ that inhibits the activation of the first rule whenever the second is activated. This priority is embedded within the rules as described earlier in this section and as a result we obtain a new first rule: $R'_1 = (\neg\alpha \wedge \beta \wedge \gamma \wedge \sim \rho, \mathbf{O}(\neg\phi))$. The implicit priority of R_4 over R_3 embeds within the latter the negation as failure obtaining a new rule $R'_3 = (\gamma \wedge \sim \sigma, \mathbf{O}(\neg\psi))$. The neural network is built from rules R'_1, R_2 and R'_3[9], notice the dotted lines in the network which are negative weighted arcs representing the negation as failures in the rules R'_1 and R'_3, with the task to inhibit the rules if the respective negation as failure given in input is activated.

[7] The connections between these input neurons and the hidden neuron of the rule represents the priorities translated with the *negation as failure*.

[8] 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.

[9] Rule with a permission $\mathbf{P}$ in the consequent are not encoded in the neural network.

5.2 Experimental results

We describe some experiments used to test the capabilities of neural networks constructed with N-CILP. We introduce the measures used to evaluate the behavior of the networks and the parameters used.

To evaluate the evaluate the performance of the neural network, we use two distinct measures: *tot* and *part*.

$$tot = \frac{\sum_{i=1}^{n} I(\bigwedge_{j=1}^{k}(c_{ij} == o_{ij}))}{n}$$
$$part = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k} I(c_{ij} == o_{ij})}{n * k}$$

n refers to the cardinality of the test set and k indicates the number of output neurons of the neural network. o_{ij} indicates the value of j-th output of the NN for the i-th test instance. c_{ij} indicates the true value (desired value) of the j-th literal of the i-th test instance. $I(\cdot)$ is the indicator, a function returning 1 if the argument is true and zero otherwise. The *tot* measure evaluates how many instances were processed entirely correctly. Instead *part* considers the number of single output neurons correctly activated.

By having 16 output neurons in the neural networks used in the test, using only *tot* to measure the accuracy could be misleading. To clarify this point we can consider an example. We can assume that by processing two instances, the neural network have produced for the first, 15 correct outputs out of the total 16. For the second it managed to return all the correct outputs. If we take into account the *tot* measure, we obtain an accuracy of 50% that does not seems a great result. Instead by considering the *part* measure, we obtain an accuracy higher than the 96%. Which better underlines that the network only missed one output out of 32 produced for the two instances given.

In our experiments we train the neural network using a *10fold cross validation*. We divide the initial data set of instances in ten distinct subsets. Each subset is then used as test set while the others are used together as training set. In this way the instances seen during training are left out of the testing phase to train ten networks and the results averaged.

In all the experiments we set the training parameters for the neural networks as follows: *learning rate*: 0.8, *momentum*: 0.3 and *training cycles*: 100 [Haykin, 1999].

Non symbolic approach comparison

We compare the learning capacity of a network built with N-CILP with a non symbolic neural network. One of the well known issues of neural networks is deciding the number of neurons to use in the hidden level. To not to put the non symbolic neural network in excessive disadvantage, we decided to adopt the same number of hidden neurons for both networks[10]. The difference between the networks involved in this test lies in their connection weights. The neural network built with N-CILP sets its weights according to the rules in the knowledge base. Instead the non symbolic network has its weights randomly initialized. One advantage of a network built with N-CILP is that even without any training, it is capable to correctly process instances by applying the rules contained in the knowledge base.

The network built with N-CILP uses a starting knowledge base containing 20 rules. During the training phase the network tries to learn 9 additional rules from the instances provided. The non symbolic network during the training phase is provided with the same instances, the difference is that this network have to learn all the 29 rules applied in the instances.

The results from the experiments show that the non symbolic neural network obtains the following accuracies: *tot*: 5,13% *part*: 45,25%. Instead the network built N-CILP: *tot*: 5,38% *part*: 49,19%. We can see that under exactly the same conditions, N-CILP improves the training-set performance of the network.

Enhancing the knowledge base

The second experiment measures how the neural network performs by increasing the number of rules in the knowledge base. This test is important because the goal of a *Neural-Symbolic System*, is not only to construct a neural network capable to compute the same semantics as rule into the knowledge base. Another important objective is to exploit the learning capabilities of the neural networks, allowing the robot to increase the number of rules in its knowledge base from what it learned[d'Avila Garcez *et al.*, 2002].

The test is done incrementally. From the full set of 29 rules, the experiment first step starts with a knowledge base containing 20 rules and tries to learn the remaining 9. Successively 2 rules are incrementally added into the initial knowledge base during each step. In this way the unknown rules that the network has to learn decreases by 2 each step. In example at the second step of the experiment the starting knowledge base contains 22 rules and the network tries to learn 7 rules during the training phase.

During each step the neural network is tested over instances where the full set of rules is applied. In this way the network continues to process using the rules already known, reducing the risk to forget them and in the meantime it tries to learn of the unknown rules.

The results of this experiment are shown in Figure 4. We can see that for the first two steps of the experiment the accuracies measured quite low. instead for the last two steps the performance of the neural network increases, reaching an accuracy peak of 98,01% for the *part* measure and 91,18% for the *tot*.

From the experiment proposed we observed a direct correlation between the number of the rules in the starting knowledge base and the performance of the neural network. Another thing that can be noticed is that the smaller becomes the number of rules that the network does not know, w.r.t. the number of rules in the initial knowledge base can impact the performances of the network, also due to the fact that a network built from a larger knowledge base possesses more connections.

6 Conclusion

In this paper we presented a way to combine a connectionist and a symbolic approach that can be used for normative

[10]The number of hidden neurons to use in the neural networks is equal to the number of rules used for the network construction with N-CILP.

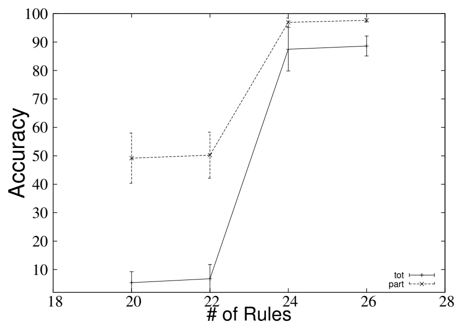

Figure 4: Accuracy of *tot* and *part* measures increasing the number of rules

reasoning. In this way agents behaving in normative environments, are able to adapt themselves to the normative evolution of the world. An important step that has not been covered by this paper concerns rules extraction. Rules extraction refers to the process where a new knowledge base is recompiled from the trained network. A method to achieve this task has already been proposed in [d'Avila Garcez *et al.*, 2002].

For a normative agent is important to be able to cope with normative problems. Here we have show how the priorities, used to achieve this task, can be embedded within the rules and translated using the N-CILP algorithm.

In the paper we provided some of the results obtained with the simulator by using our approach for a normative agent. We are aware that more experiments are needed in order to claim the validity of the approach. However we believe that the results obtained so far are promising. We show a comparison between our approach and a (not so disadvantaged) non symbolic neural network. Further comparisons with other approaches for dynamic normative system should be made. In example like comparing the pure symbolic approach used by [Corapi *et al.*, 2010], based on inductive learning, and our neural-symbolic approach.

A related line of research involves the area of Argumentation. Argumentation has been proposed, among other things, as a method to help symbolic machine learning. It would be interesting to investigate the links between the work presented here, argumentation applied to law, and the neural symbolic approach to argumentation introduced in [d'Avila Garcez *et al.*, 2005].

Acknowledgements

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

References

[Boella and van der Torre, 2006] Guido Boella and Leendert van der Torre. A game theoretic approach to contracts in multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 36(1):68–79, 2006.

[Boella *et al.*, 2009] Guido Boella, Gabriella Pigozzi, and Leendert van der Torre. Normative framework for normative system change. In *8th Int. Joint Conf. on Autonomous Agents and Multiagent Systems AAMAS 2009*, pages 169–176. IFAAMAS, 2009.

[Boella *et al.*, 2010] Guido Boella, Silvano Colombo Tosatto, Artur S. d'Avila Garcez, and Valerio Genovese. On the relationship between i-o logic and connectionism. In *13th International Workshop on Non-Monotonic Reasoning*, 2010.

[Boella *et al.*, 2011] Guido Boella, Silvano Colombo Tosatto, Artur S. d'Avila Garcez, Dino Ienco, Valerio Genovese, and Leendert van der Torre. Neural symbolic systems for normative agents. In *10th International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[Corapi *et al.*, 2010] Domenico Corapi, Marina De Vos, Julian Padget, Alessandra Russo, and Ken Satoh. Norm refinement and design through inductive learning. In *11th International Workshop on Coordination, Organization, Institutions and Norms in Agent Systems COIN 2010*, pages 33–48, 2010.

[d'Avila Garcez and Zaverucha, 1999] Artur S. d'Avila Garcez and Gerson Zaverucha. The connectionist inductive learning and logic programming system. *Applied Intelligence*, 11:59–77, July 1999.

[d'Avila Garcez *et al.*, 2002] Artur S. d'Avila Garcez, Krysia B. Broda, and Dov M. Gabbay. *Neural-Symbolic Learning Systems*. Perspectives in Neural Computing. Springer, 2002.

[d'Avila Garcez *et al.*, 2005] Artur S. d'Avila Garcez, Dov M. Gabbay, and Luis C. Lamb. Value-based argumentation frameworks as neural-symbolic learning systems. *J. of Logic and Computation*, 15(6):1041–1058, 2005.

[Haykin, 1999] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

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

[Menegatti, 2007] Emanuele Menegatti. Robocup soccer humanoid league rules and setup, 2007.

[Prakken and Sergot, 1996] Henry Prakken and Marek Sergot. Contrary-to-duty obligations. *Studia Logica*, 57, 1996.

[Sen and Airiau, 2007] Sandip Sen and Stéphane Airiau. Emergence of norms through social learning. In *Procs. of the 20th International Joint Conference on Artificial Intelligence - IJCAI*, pages 1507–1512, 2007.

[Shoham and Tennenholtz, 1997] Yoav Shoham and Moshe Tennenholtz. On the emergence of social conventions: Modeling, analysis, and simulations. *Artificial Intelligence*, 94(1-2):139–166, 1997.

[Towell and Shavlik, 1994] Geoffrey G. Towell and Jude W. Shavlik. Knowledge-based artificial neural networks. *Artif. Intell.*, 70:119–165, October 1994.