A CONNECTIONIST INDUCTIVE LEARNING SYSTEM FOR MODAL LOGIC PROGRAMMING

Artur S. d'Avila Garcez$^{\alpha}$, *Luís C. Lamb*$^{\beta}$ *and Dov M. Gabbay*$^{\gamma}$

$^{\alpha}$Dept. of Computing, City University, London, EC1V 0HB, UK. aag@soi.city.ac.uk
$^{\beta}$Instituto de Informática, UFRGS, Porto Alegre, RS, 91501-970, Brazil. lamb@inf.ufrgs.br
$^{\gamma}$Dept. of Computer Science, King's College, London WC2R2LS, UK. dg@dcs.kcl.ac.uk

ABSTRACT

Neural-Symbolic integration has become a very active research area in the last decade. In this paper, we present a new massively parallel model for modal logic. We do so by extending the language of Modal Prolog [15, 18] to allow modal operators in the head of the clauses. We then use an ensemble of *C-IL²P* neural networks [7, 9] to encode the extended modal theory (and its relations), and show that the ensemble computes a fixpoint semantics of the extended theory. An immediate result of our approach is the ability to perform learning from examples efficiently using each network of the ensemble. Therefore, one can adapt the extended *C-IL²P* system by training possible world representations.

Keywords: Hybrid Systems, Neural-Symbolic Integration, Modal Logic, Change of Representation.

1. INTRODUCTION

Neural-Symbolic integration concerns the utilization of problem-specific symbolic knowledge within the neurocomputing paradigm [7]. In spite of the progress in the last decade, neural-symbolic systems have not been shown to fully represent and learn first order logic [5]. We believe that a promising approach to unravel this problem is to investigate ways of representing and learning the necessity and possibility operators, and the acessibility relation of propositional modal logic [3] in neural networks[1].

In this paper, we propose a new massively parallel model for propositional modal logic, thus contributing to the representation of quantification in neural networks. We present an algorithm to translate a modal theory into an ensemble of *Connectionist Inductive Learning and Logic Programming (C-IL²P)* networks [7], and show that the ensemble computes a fixpoint semantics of the theory. As a result, the ensemble can be seen as a new massively parallel model for modal logic. In addition, since each *C-IL²P* network can be trained efficiently using the *Backpropagation* learning algorithm [17], one can adapt the *C-IL²P* ensemble by training each network with examples.

In Section 2, we briefly present the basic concepts of modal logic used in this paper. In Section 3, we present a Modalities Algorithm that translates extended modal logic programs into ensembles of *C-IL²P* networks. We then show that the ensemble computes a fixpoint semantics of the given modal program, thus proving the correctness of the Modalities Algorithm. In Section 4, we conclude and discuss directions for future work.

2. EXTENDED MODAL LOGIC PROGRAMS

In this section, we present some basic concepts of modal logic. We also extend modal Prolog [18] to allow the necessity ($\Box$) and possibility ($\Diamond$) modalities to appear in the head of the clauses, as well as default negation ($\sim$) to appear in the body of the clauses [4]. Finally, we define a fixpoint semantics for such extension, to be computed by our ensemble of *C-IL²P* networks. As usual, we assume that any clause is ground over a finite domain.

A main feature of modal logics is the use of *possible world semantics*, which has significantly contributed to the development of new non-classical logics, many of which with great impact in computing science. A proposition is necessary ($\Box$) in a world if it is true in all worlds which are possible in relation to that world, whereas it is possible ($\Diamond$) in a world if it is true in at least one world which is possible in relation to that same world.

Definition 1 *A* modal atom *is of the form MA where $M \in \{\Box, \Diamond\}$ and A is an atom. A* modal literal *is of the form ML where L is a literal.*

Definition 2 *A* modal prolog program *is a finite set of clauses of the form $MA_1, ..., MA_n \rightarrow A$.*

We define *extended modal programs* as modal prolog programs extended with modalities $\Box$ and $\Diamond$ in the head of clauses, and default negation $\sim$ in the body of clauses. In

[1]Modal logic was found to be appropriate to study mathematical necessity (in the logic of provability), time, knowledge, belief, obligation and other concepts and modalities. In artificial intelligence and computing, modal logics are among the most employed formalisms to analyse and represent multi-agent systems and concurrency properties [10].

addition, each clause is labelled by the possible world in which they hold, similarly to Gabbay's Labelled Deductive Systems [12], as follows.

Definition 3 *An* extended modal program *is a finite set of clauses C of the form $\omega_i : ML_1, ..., ML_n \rightarrow MA$, where ω_i is a label representing a world in which the associated clause holds, and a finite set of relations $\mathcal{R}(\omega_i, \omega_j)$ between worlds ω_i and ω_j in C.*

For example: $\mathcal{P} = \{\omega_1 : r \rightarrow \Box q, \omega_1 : \Diamond s \rightarrow r, \omega_2 : s, \omega_3 : q \rightarrow \Diamond p, \mathcal{R}(\omega_1, \omega_2), \mathcal{R}(\omega_1, \omega_3)\}$ is an extended modal program. The $\Box$ and $\Diamond$ modalities will have the following interpretation.

Definition 4 (Kripke Models for Modal Logic) *Let $\mathcal{L}$ be a modal language. A* model *for $\mathcal{L}$ is a tuple $\mathcal{M} = \langle \Omega, \mathcal{R}, v \rangle$ where Ω is a set of possible worlds, v is a mapping that assigns to each propositional letter of $\mathcal{L}$ a subset of Ω, and $\mathcal{R}$ is a binary relation over Ω, such that: (i) $(\mathcal{M}, \omega) \models \Box\alpha$ iff for all $\omega_1 \in \Omega$, if $\mathcal{R}(\omega, \omega_1)$ then $(\mathcal{M}, \omega_1) \models \alpha$, and (ii) $(\mathcal{M}, \omega) \models \Diamond\alpha$ iff there exists a ω_1 such that $\mathcal{R}(\omega, \omega_1)$ and $(\mathcal{M}, \omega_1) \models \alpha$.*

In what follows, we define a model-theoretic semantics for extended modal programs. When computing the semantics of the program, we have to consider both the fixpoint of a particular world, and the fixpoint of the program as a whole. When computing the fixpoint in each world, we have to consider the consequences derived locally and the consequences derived from the interaction between worlds. Locally, fixpoints are computed as in the stable model semantics of logic programming, by simply renaming each modal literal ML_i by a new literal L_j not in the language $\mathcal{L}$, and applying the Gelfond-Lifschitz Transformation [2] to it. When considering interacting worlds, there are two cases to be addressed, according to the $\Box$ and $\Diamond$ modalities, and the acessibility relation $\mathcal{R}$, which might render additional consequences in each world.

Definition 5 (Modal Immediate Consequence Operator) *Let $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_k\}$ be an extended modal program, where each $\mathcal{P}_i$ is the set of modal clauses that hold in a world ω_i $(1 \leq i \leq k)$. Let $B_\mathcal{P}$ be the Herbrand base of $\mathcal{P}$ and I be a Herbrand interpretation for $\mathcal{P}_i$. The mapping $MT_{\mathcal{P}_i} : 2^{B_\mathcal{P}} \rightarrow 2^{B_\mathcal{P}}$ in ω_i is defined as follows: $MT_{\mathcal{P}_i}(I) = \{MA \in B_\mathcal{P} \mid$ either (i) or (ii) or (iii) below holds$\}$. (i) $ML_1, ..., ML_n \rightarrow MA$ is a clause in $\mathcal{P}_i$ and $\{ML_1, ..., ML_n\} \subseteq I$; (ii) $M = \Diamond$ and there exists a world ω_j such that $\mathcal{R}(\omega_i, \omega_j)$, $ML_1, ..., ML_m \rightarrow A$ is a clause in $\mathcal{P}_j$ and $\{ML_1, ..., ML_m\} \subseteq J$, where J is a Herbrand interpretation for $\mathcal{P}_j$; (iii) $M = \Box$ and for each world ω_j such that $\mathcal{R}(\omega_i, \omega_j)$, $ML_1, ..., ML_o \rightarrow A$ is a clause in $\mathcal{P}_j$ and $\{ML_1, ..., ML_o\} \subseteq K$, where K is a Herbrand interpretation for $\mathcal{P}_j$.*

Definition 6 (Global Modal Immediate Consequence Operator) *Let $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_k\}$ be an extended modal program. Let $B_\mathcal{P}$ be the Herbrand base of $\mathcal{P}$ and I_i be a Herbrand interpretation for $\mathcal{P}_i$ $(1 \leq i \leq k)$. The mapping $MT_\mathcal{P} : 2^{B_\mathcal{P}} \rightarrow 2^{B_\mathcal{P}}$ is defined as follows: $MT_\mathcal{P}(I_1, ..., I_k) = \bigcup_{j=1}^{k}\{MT_{\mathcal{P}_j}\}$.*

In the case of definite extended modal programs, by renaming each modal atom MA_i by a new atom A_j, we can apply the following result of Ramanujam [16], regarding the fixpoint semantics of distributed definite logic programs.

Theorem 1 (Minimal Model of Distributed Programs [16]) *For each distributed definite logic program $\mathcal{P}$, the function $MT_\mathcal{P}$ has a unique fixpoint. The sequence of all*

$$MT_\mathcal{P}^m(I_1, ..., I_k), m \in \mathbb{N},$$

converges to this fixpoint $MT_\mathcal{P}^\varpi(I_1, ..., I_k)$, for each $I_i \subseteq 2^{B_\mathcal{P}}$.

In order to provide a fixpoint semantics for extended modal programs, we have to extend the definition of *acceptable* logic programs [1].

Definition 7 (Level Mapping) *Let $\mathcal{P}$ be a general logic program. A* level mapping *for $\mathcal{P}$ is a function $|\ \ | : B_\mathcal{P} \rightarrow \mathbb{N}$ from ground atoms to natural numbers. For $A \in B_\mathcal{P}$, $|A|$ is called the* level *of A and $|\sim A| = |A|$.*

Definition 8 (Acceptable Programs) *Let $\mathcal{P}$ be a program, $|\ \ |$ a level mapping for $\mathcal{P}$, and $\mathbf{I}$ a model of $\mathcal{P}$. $\mathcal{P}$ is called* acceptable *w. r. t $|\ \ |$ and $\mathbf{I}$ if for every clause $L_1, ..., L_k \rightarrow A$ in $\mathcal{P}$ the following implication holds. If $\mathbf{I} \models \bigwedge_{j=1}^{i-1} L_j$ then $|A| > |L_j|$ for $1 \leq i \leq k$.*

Theorem 2 (Minimal Model of Acceptable Programs [11]) *For each acceptable program $\mathcal{P}$, the function $T_\mathcal{P}$ has a unique fixpoint[2]. The sequence of all $T_\mathcal{P}^m(I), m \in \mathbb{N}$, converges to this fixpoint $T_\mathcal{P}^\varpi(I)$ (which is identical to the* stable model *of $\mathcal{P}$ [14])[3], for each $I \subseteq B_\mathcal{P}$.*

Definition 9 (Acceptable Extended Modal Programs) *An extended modal program $\mathcal{P}$ is acceptable iff the program $\mathcal{P}'$ obtained by renaming each modal literal ML_i in $\mathcal{P}$ by a new literal L_j not in the language $\mathcal{L}$ is acceptable.*

Clearly, the classical results about the fixpoint semantics of logic programming also apply to extended modal programs. The proof of Theorem 3, below, follows directly from Theorems 2 and 1.

[2]The mapping $T_\mathcal{P}$ is defined as follows: Let I be a Herbrand *interpretation,* then $T_\mathcal{P}(I) = \{A_0 \mid L_1, ..., L_n \rightarrow A_0$ is a clause in $\mathcal{P}$ and $\{L_1, ..., L_n\} \subseteq I\}$.

[3]An interpretation I of a general logic program $\mathcal{P}$ is called stable iff $T_{\mathcal{P}_I}(I) = I$, where $\mathcal{P}_I$ is the definite program obtained by applying the Gelfond-Lifschitz Transformation on $\mathcal{P}$.

Theorem 3 (Minimal Model of Acceptable Extended Modal Programs) *For each acceptable extended modal program $\mathcal{P}$, the function $MT_{\mathcal{P}}$ has a unique fixpoint. The sequence of all $MT_{\mathcal{P}}^{m}(I_1, ..., I_k), m \in \mathbb{N}$, converges to this fixpoint $MT_{\mathcal{P}}^{\varpi}(I_1, ..., I_k)$, for each $I_i \subseteq 2^{B_{\mathcal{P}}}$.*

Finally, note that in the above semantics, the choice of an arbitrary world for $\Diamond$ elimination (made before the computation of $MT_{\mathcal{P}}$) may lead to different fixpoints of a given extended modal program. Such a choice is similar to the approach adopted by Gabbay in [13], in which one chooses a point in the future for execution and then backtracks if judged necessary (and at all possible).

3. CONNECTIONIST MODAL PROGRAMS

In this section, we present a new *Modalities Algorithm* that translates extended modal programs into neural networks ensembles. By using an ensemble of *C-IL^{2}P* networks, we can enhance the expressive power of the *C-IL^{2}P* system [7], yet maintaining the simplicity needed for performing inductive learning efficiently. Let us first recall how *C-IL^{2}P* works.

C-IL^{2}P [7] is a massively parallel computational model based on artificial neural networks that integrates inductive learning from examples and background knowledge with deductive learning from logic programming. A *Translation Algorithm* maps a general logic program $\mathcal{P}$ into a single hidden layer neural network $\mathcal{N}$ such that $\mathcal{N}$ computes the least fixpoint of $\mathcal{P}$. This provides a massively parallel model for computing the stable model semantics of $\mathcal{P}$. In addition, $\mathcal{N}$ can be trained with examples using *Backpropagation*, having $\mathcal{P}$ as background knowledge. The knowledge acquired by training can then be extracted, closing the learning cycle [6].

Let us exemplify how the *Translation Algorithm* works. Each rule (r_l) of $\mathcal{P}$ is mapped from the input layer to the output layer of $\mathcal{N}$ through one neuron (N_l) in the single hidden layer of $\mathcal{N}$. Intuitively, the *Translation Algorithm* from $\mathcal{P}$ to $\mathcal{N}$ has to implement the following conditions: (**C1**) The input potential of a hidden neuron (N_l) can only exceed N_l's threshold (θ_l), activating N_l, when all the positive antecedents of r_l are assigned the truth-value $true$ while all the negative antecedents of r_l are assigned $false$; and (**C2**) The input potential of an output neuron (A) can only exceed A's threshold (θ_A), activating A, when at least one hidden neuron N_l that is connected to A is activated.

Example 1 *Consider the logic program $\mathcal{P} = \{B, C, \sim D \rightarrow A; E, F \rightarrow A; \rightarrow B\}$. The* Translation Algorithm *derives the network $\mathcal{N}$ of Figure 1, setting weights ($W's$) and thresholds ($\theta's$) in such a way that conditions (**C1**) and (**C2**) above are satisfied. Note that, if $\mathcal{N}$ ought to be fully-connected, any other link (not shown in Figure 1) should receive weight zero initially.*

Note that, in Example 1, each input and output neuron of $\mathcal{N}$ is associated with an atom of $\mathcal{P}$. As a result, each input and output vector of $\mathcal{N}$ can be associated with an interpretation for $\mathcal{P}$. Note also that each hidden neuron N_l corresponds to a rule r_l of $\mathcal{P}$. In order to compute the stable models of $\mathcal{P}$, output neuron B should feed input neuron B such that $\mathcal{N}$ is used to iterate $T_{\mathcal{P}}$, the fixpoint operator of $\mathcal{P}$. $\mathcal{N}$ will eventually converge to a stable state which is identical to a stable model of $\mathcal{P}$ (*C-IL^{2}P's Translation Algorithm* and the proof of convergence are given in [9]).

In order to translate extended modal programs into neural networks ensembles, we use *C-IL^{2}P's Translation Algorithm* for creating each network of the ensemble, and the new *Modalities Algorithm* (below) for interconnecting the different networks. Figure 2 shows an ensemble of three *C-IL^{2}P* networks, representing possible worlds ω_1, ω_2 and ω_3, which might communicate in different ways. Due to the simplicity of each *C-IL^{2}P* network, performing inductive learning within each *possible world* (e.g., ω_1) seems straightforward. The problem to be tackled here is, therefore, how to learn (or set up) the connections that establish the necessary communication between networks (e.g., ω_1 and ω_2). In the case of modal logic, such connections are defined analogously to the modal rules of natural deduction (Table 1). For example, we know that if $\Box A$ is $true$ in ω_1 then A must be $true$ in ω_2, whenever $\mathcal{R}(\omega_1, \omega_2)$. As a result, we must connect $\Box A$ in the output layer of ω_1 to A in the output layer of ω_2, making sure that whenever $\Box A$ is activated in ω_1, A is activated in ω_2[4].

$$\frac{\omega_1{:}\Box\alpha, \mathcal{R}(\omega_1,\omega_2)}{\omega_2{:}\alpha}\ \Box E \qquad \frac{\omega_1{:}\Diamond\alpha}{\omega_2{:}\alpha, \mathcal{R}(\omega_1,\omega_2)}\ \Diamond E$$

$$\frac{[\omega_i{:}\alpha]\ldots\mathcal{R}(\omega_1,\omega_i)}{\omega_1{:}\Box\alpha}\ \Box I \qquad \frac{\omega_2{:}\alpha, \mathcal{R}(\omega_1,\omega_2)}{\omega_1{:}\Diamond\alpha}\ \Diamond I$$

Table 1: Modal rules of natural deduction[5].

Modalities Algorithm:

1. Let $\mathcal{P}$ be an extended modal program with clauses of the form $\omega_i : ML_1, ..., ML_k \rightarrow MA$ $(1 \leq i \leq n)$, where L_j $(0 \leq j \leq k)$ is a literal, A is an atom and $M \in \{\Box, \Diamond\}$. As in the case of individual *C-IL^{2}P* networks, we start by calculating $MAX_{\mathcal{P}}(k_1, ..., k_q,\ \mu_1,\ ..., \mu_q, n)$, which returns the greatest element among k_l, μ_l $(1 \leq l \leq q)$ and n, where k_l is the number of literals in the body of rule r_l in $\mathcal{P}$, μ_l is the number of rules with the same atom in the head for each

[4]Note that input and output neurons may represent literals $\Box L$, $\Diamond L$ or L.

[5]The $\Diamond E$ rule can be seen as a *skolemization* of the existential quantifier over possible worlds, which is semantically implied by the formula $\Diamond\alpha$ in the premise. In the $\Box I$ rule, the temporary assumption should be read as "given an arbitrary accessible world ω_i". The rule of $\Diamond I$ represents that if we have a relation $\mathcal{R}(\omega_1, \omega_2)$, and if α holds at ω_2 then it must be the case that α holds at ω_1. The rule $\Box E$ represents that if $\Box\alpha$ holds at a world ω_1, and ω_1 is related to ω_2, then we can infer that α holds at ω_2.

rule r_l in $\mathcal{P}$, and n is the number of *C-IL²P* networks (possible worlds) in the ensemble.

2. Calculate $A_{min} > \frac{MAX_{\mathcal{P}}(k_1,\ldots,k_q,\mu_1,\ldots,\mu_q,n)-1}{MAX_{\mathcal{P}}(k_1,\ldots,k_q,\mu_1,\ldots,\mu_q,n)+1}$, which will denote the minimum activation for a neuron to be considered active (in which case its associated atom will be considered $true$).

3. Let $\mathcal{P}_i \subseteq \mathcal{P}$ be the set of clauses labelled by ω_i in $\mathcal{P}$. Let $\mathcal{N}_i$ be the neural network that denotes $\mathcal{P}_i$. Let $W^M \in \Re$ be such that $W^M > h^{-1}(A_{\min}) + \mu_l W + \theta_A$, where $h(x) = \frac{2}{1+e^{-x}} - 1$,[6] $W \geq 2 \cdot \frac{\ln(1+A_{\min})-\ln(1-A_{\min})}{MAX_{\mathcal{P}}(k_1,\ldots,k_q,\mu_1,\ldots,\mu_q)\cdot(A_{\min}-1)+A_{\min}+1}$, and $\theta_A = \frac{(1+A_{\min})(1-\mu_l)}{2} \cdot W$ are obtained from *C-IL²P's Translation Algorithm*[7].

4. For each $\mathcal{P}_i$ do:

   (a) Rename each ML_j in $\mathcal{P}_i$ by a new literal not occuring in $\mathcal{P}$ of the form $L_j^{\Box}$ if $M = \Box$, or $L_j^{\Diamond}$ if $M = \Diamond$;[8] (b) Call *C-IL²P's Translation Algorithm;*[9]

5. For each output neuron $L_j^{\Diamond}$ in $\mathcal{N}_i$, do:

   (a) Add a hidden neuron L_j^M to an arbitrary network $\mathcal{N}_k$ $(0 \leq k \leq n)$ in the ensemble such that $\mathcal{R}(\omega_i, \omega_k)$; (b) Set the step function $y = s(x)$; where $y = 1$ if $x > 0$, and $y = 0$ otherwise; as the activation function of L_j^M; (c) Connect $L_j^{\Diamond}$ in $\mathcal{N}_i$ to L_j^M and set the connection weight to 1; (d) Set the threshold θ^M of L_j^M such that $-1 < \theta^M < A_{min}$; and (e) Connect L_j^M to L_j in $\mathcal{N}_k$ and set the connection weight to W^M.

6. For each output neuron $L_j^{\Box}$ in $\mathcal{N}_i$, do:

   (a) Add a hidden neuron L_j^M to each network $\mathcal{N}_k$ $(0 \leq k \leq n)$ such that $\mathcal{R}(\omega_i, \omega_k)$; (b) Set $s(x)$ as the activation function of L_j^M; (c) Connect $L_j^{\Box}$ in $\mathcal{N}_i$ to L_j^M and set the connection weight to 1; (d) Set the threshold θ^M of L_j^M such that $-1 < \theta^M < A_{min}$; and (e) Connect L_j^M to L_j in $\mathcal{N}_k$ and set the connection weight to W^M.

7. For each output neuron L_j in $\mathcal{N}_k$ such that $\mathcal{R}(\omega_i, \omega_k)$ $(0 \leq i \leq m)$, do:

   (a) Add a hidden neuron $L_j^{\vee}$ to $\mathcal{N}_i$; (b) Set $s(x)$ as the activation function of $L_j^{\vee}$; and (c) For each output neuron $L_j^{\Diamond}$ in $\mathcal{N}_i$, do:

   (i) Connect L_j in $\mathcal{N}_k$ to $L_j^{\vee}$ and set the connection weight to 1; (ii) Set the threshold $\theta^{\vee}$ of $L_j^{\vee}$ such that $-nA_{min} < \theta^{\vee} < A_{min} - (n-1)$; and (iii) Connect $L_j^{\vee}$ to $L_j^{\Diamond}$ in $\mathcal{N}_i$ and set the connection weight to W^M.

8. For each output neuron L_j in $\mathcal{N}_k$ such that $\mathcal{R}(\omega_i, \omega_k)$ $(0 \leq i \leq m)$, do:

   (a) Add a hidden neuron $L_j^{\wedge}$ to $\mathcal{N}_i$; (b) Set $s(x)$ as the activation function of $L_j^{\wedge}$; and (c) For each output neuron $L_j^{\Box}$ in $\mathcal{N}_i$, do:

   (i) Connect L_j in $\mathcal{N}_k$ to $L_j^{\wedge}$ and set the connection weight to 1; (ii) Set the threshold $\theta^{\wedge}$ of $L_j^{\wedge}$ such that $n-(1+A_{min}) < \theta^{\wedge} < nA_{min}$; and (iii) Connect $L_j^{\wedge}$ to $L_j^{\Box}$ in $\mathcal{N}_i$ and set the connection weight to W^M.

Let us now illustrate the use of the *Modalities Algorithm* with the following example.

Example 2 *Let $\mathcal{P} = \{\omega_1 : r \rightarrow \Box q, \omega_1 : \Diamond s \rightarrow r, \omega_2 : s, \omega_3 : q \rightarrow \Diamond p, \mathcal{R}(\omega_1,\omega_2), \mathcal{R}(\omega_1,\omega_3)\}$. We start by applying* C-IL²P's Translation Algorithm, *which creates three neural networks to represent the worlds ω_1, ω_2, and ω_3 (see Figure 3). Then, we apply the* Modalities Algorithm. *Hidden neurons labelled by $\{M, \vee, \wedge\}$ are created using the* Modalities Algorithm. *The remaining neurons are all created using the* Translation Algorithm. *For the sake of clarity, unconnected input and output neurons are not shown in Figure 3.*

Taking $\mathcal{N}_1$ (which represents ω_1), output neurons $L_j^{\Diamond}$ should be connected to output neurons L_j in an arbitrary network $\mathcal{N}_i$ (which represents ω_i) to which $\mathcal{N}_1$ is related. For example, taking $\mathcal{N}_i = \mathcal{N}_2$, $\Diamond s$ in $\mathcal{N}_1$ is connected to s in $\mathcal{N}_2$. Then, output neurons $L_j^{\Box}$ should be connected to output neurons L_j in every network $\mathcal{N}_i$ to which $\mathcal{N}_1$ is related. For example, $\Box q$ in $\mathcal{N}_1$ is connected to q in both $\mathcal{N}_2$ and $\mathcal{N}_3$. Now, taking $\mathcal{N}_2$, output neurons L_j need to be connected to output neurons $L_j^{\Diamond}$ and $L_j^{\Box}$ in every world $\mathcal{N}_j$ related to $\mathcal{N}_2$. For example, s in $\mathcal{N}_2$ is connected to $\Diamond s$ in $\mathcal{N}_1$ via the hidden neuron denoted by $\vee$ in Figure 3, while q in $\mathcal{N}_2$ is connected to $\Box q$ in $\mathcal{N}_1$ via the hidden neuron denoted by $\wedge$. Similarly, q in $\mathcal{N}_3$ is connected to $\Box q$ in $\mathcal{N}_1$ via $\wedge$. The algorithm terminates when all output neurons have been connected.

We are now in a position to show that the ensemble of neural networks $\mathcal{N}$ obtained from the above *Modalities Algorithm* is equivalent to the original extended modal program $\mathcal{P}$, in the sense that $\mathcal{N}$ computes the *modal immediate consequence operator $MT_{\mathcal{P}}$* of $\mathcal{P}$ (see Definition 5).

Theorem 4 *For any extended modal program $\mathcal{P}$ there exists an ensemble of single hidden layer neural networks $\mathcal{N}$ such that $\mathcal{N}$ computes the modal fixpoint operator $MT_{\mathcal{P}}$ of $\mathcal{P}$.*

[6] We use the bipolar sigmoid function $h(x)$ as activation function.

[7] Recall that μ_l is the number of connections to output neuron l (see Example 1).

[8] This labelling of neurons allows us to treat each modal literal ML_j as a literal L_j and apply *C-IL²P's Translation Algorithm* directly to $\mathcal{P}_i$.

[9] This is done to construct the *C-IL²P* network that represents each possible world ω_i.

Proof. *We have to show that there exists $W > 0$ such that the network $\mathcal{N}$, obtained by the above* Modalities Algorithm, *computes $MT_{\mathcal{P}}$. Assume that $\mathcal{N}_i$ and $\mathcal{N}_j$ are two arbitrary sub-networks of $\mathcal{N}$, representing possible worlds ω_i and ω_j, respectively, such that $\mathcal{R}(\omega_i, \omega_j)$. We distinguish two cases: (a) rules with modalities $\Box$ and $\Diamond$ in the head, and (b) rules with no modalities in the head. (a) Firstly, note that rules with $\Box$ in the head must satisfy $\Box E$, while rules with $\Diamond$ in the head must satisfy $\Diamond E$ (see Table 1). Given input vectors* **i** *and* **j** *to $\mathcal{N}_i$ and $\mathcal{N}_j$, respectively, each neuron A in the output layer of $\mathcal{N}_j$ is active ($A > A_{min}$) if and only if: (i) there exists a clause of $\mathcal{P}_j$ of the form $ML_1, ..., ML_k \rightarrow A$ s.t. $ML_1, ..., ML_k$ are satisfied by interpretation* **j**, *or (ii) there exists a clause of $\mathcal{P}_i$ of the form $ML_1, ..., ML_k \rightarrow \Box A$ s.t. $ML_1, ..., ML_k$ are satisfied by interpretation* **i**, *or even (iii) there exists a clause of $\mathcal{P}_i$ of the form $ML_1, ..., ML_k \rightarrow \Diamond A$ s.t. $ML_1, ..., ML_k$ are satisfied by interpretation* **i**, *and the* Modalities Algorithm *(step 5) has selected $\mathcal{N}_j$ as the arbitrary network $\mathcal{N}_k$. (b) Rules with no modalities must satisfy $\Box I$ and $\Diamond I$ in Table 1. Given input vectors* **i** *and* **j** *to $\mathcal{N}_i$ and $\mathcal{N}_j$, respectively, each neuron $\Box A$ in the output layer of $\mathcal{N}_i$ is active ($\Box A > A_{min}$) if and only if: (i) there exists a clause of $\mathcal{P}_i$ of the form $ML_1, ..., ML_k \rightarrow \Box A$ s.t. $ML_1, ..., ML_k$ are satisfied by interpretation* **i**, *or (ii) for all $\mathcal{N}_j$, there exists a clause of $\mathcal{P}_j$ of the form $ML_1, ..., ML_k \rightarrow A$ s.t. $ML_1, ..., ML_k$ are satisfied by interpretation* **j**. *Each neuron $\Diamond A$ in the output layer of $\mathcal{N}_i$ is active ($\Diamond A > A_{min}$) if and only if: (iii) there exists a clause of $\mathcal{P}_i$ of the form $ML_1, ..., ML_k \rightarrow \Diamond A$ s.t. $ML_1, ..., ML_k$ are satisfied by interpretation* **i**, *or (iv) there exists a clause of $\mathcal{P}_j$ of the form $ML_1, ..., ML_k \rightarrow A$ s.t. $ML_1, ..., ML_k$ are satisfied by interpretation* **j**. *The complete proof, in which cases $a(i)$ to $a(iii)$ and $b(i)$ to $b(iv)$ are proved, is given in [8].* $\Box$

Now, if we make output neurons feed their corresponding input neurons in each (sub)network, the resulting (partially recurrent) ensemble $\mathcal{N}^r$ can be used to iterate $MT_{\mathcal{P}}$ in parallel, as in the case of *C-IL²P* (but now to compute a modal program). For example, in Figure 3, if we connect output neurons $\Diamond s$ and r to input neurons $\Diamond s$ and r, respectively, in $\mathcal{N}_1$, and output neuron q to input neuron q in $\mathcal{N}_3$, the ensemble computes $\{\Diamond s, r, \Box q\}$ in ω_1, $\{s, q\}$ in ω_2, and $\{q, \Diamond s\}$ in ω_3.[10] As expected, these are some of the logical consequences of the program $\mathcal{P}$ given in Example 2.

Corollary 5 *Let $\mathcal{P}$ be an acceptable extended modal program. There exists an ensemble of single hidden layer neural networks $\mathcal{N}^r$ such that, starting from an arbitrary initial input, $\mathcal{N}^r$ converges to a stable state and yields the unique fixpoint ($MT_{\mathcal{P}}^{\varpi}(I)$) of $MT_{\mathcal{P}}$.*

Proof. *Assume that $\mathcal{P}$ is an acceptable program. By Theorem 4, $\mathcal{N}^r$ computes $MT_{\mathcal{P}}$. Recurrently connected, $\mathcal{N}^r$ computes the upwards powers ($T_{\mathcal{P}}^m(I)$) of $T_{\mathcal{P}}$. Finally, by Theorem 3, $\mathcal{N}^r$ computes the unique fixpoint ($MT_{\mathcal{P}}^{\varpi}(I)$) of $MT_{\mathcal{P}}$.* $\Box$

[10]Although the computation is done in parallel in $\mathcal{N}$, following it by starting from facts (such as s in ω_2) would help in verifying this.

Hence, in order to use $\mathcal{N}$ as a massively parallel model for modal logic, we simply need to recurrently connect its (sub)networks $\mathcal{N}_i$ using fixed-weight links $W_r = 1$.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a new massively parallel model for modal logic, thus contributing towards the representation of quantification in neural networks. We have defined an extension of modal logic programming [18], which allows modal operators in the head of clauses. We then presented an algorithm to translate the modal theory into an ensemble of *C-IL²P* neural networks [7], and showed that the ensemble computes a fixpoint semantics of the theory. As a result, the ensemble can be seen as a new massively parallel model for modal logic. In addition, since each *C-IL²P* network can be trained efficiently using *Backpropagation* [9], one can adapt the *C-IL²P* ensemble by training possible world representations from examples in each network.

Our next step is to perform experiments on learning possible world representations in the *C-IL²P* ensemble. This would lead us to another interesting avenue of research, namely, rule extraction from neural networks ensembles, which would need to consider extraction methods for more expressive knowledge representation formalisms [6]. In addition, extensions of the basic modal *C-IL²P* ensemble presented in this paper include the study of how to represent properties of other modal logics, e.g., S4 and S5, and of inference and learning of fragments of first order logic. Finally, the addition of modalities to the *C-IL²P* system leads us towards richer distributed knowledge representation and learning mechanisms, with a broader range of potential applications, including practical reasoning and learning in a multiagent environment [10].

5. REFERENCES

[1] K. R. Apt and D. Pedreschi. Reasoning about termination of pure prolog programs. *Information and Computation*, 106:109–157, 1993.

[2] G. Brewka and T. Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109:297–356, 1999.

[3] A. Chagrov and M. Zakharyaschev. *Modal Logic*. Clarendon Press, Oxford, 1997.

[4] K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322, Plenum Press, New York, 1978.

[5] I. Cloete and J. M. Zurada, editors. *Knowledge-Based Neurocomputing*. The MIT Press, 2000.

[6] A. S. d'Avila Garcez, K. Broda, and D. M. Gabbay. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence*, 125:155–207, 2001.

[7] A. S. d'Avila Garcez, K. Broda, and D. M. Gabbay. *Neural-Symbolic Learning Systems: Foundations and Applications*. Perspectives in Neural Computing. Springer-Verlag, 2002.

[8] A. S. d'Avila Garcez, L. C. Lamb, and D. M. Gabbay. A connectionist inductive learning system for modal logic programming. Technical Report 2002/6, Department of Computing, Imperial College, London, 2002.

[9] A. S. d'Avila Garcez and G. Zaverucha. The connectionist inductive learning and logic programming system. *Applied Intelligence Journal, Special Issue on Neural Networks and Structured Knowledge*, 11(1):59–77, 1999.

[10] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[11] M. Fitting. Metric methods: Three examples and a theorem. *Journal of Logic Programming*, 21:113–127, 1994.

[12] D. M. Gabbay. *Labelled Deductive Systems*. Clarendom Press.

[13] D. M. Gabbay. The declarative past and imperative future. In H. Barringer, editor, *Proceedings of the Colloquium on Temporal Logic and Specifications*, LNCS 398. Springer-Verlag, 1989.

[14] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the fifty Logic Programming Symposium*, 1988.

[15] M. A. Orgun and W. Ma. An overview of temporal and modal logic programming. In *Proceedings of International Conference on Temporal Logic, ICTL'94*, LNAI 827, pages 445–479. Springer.

[16] R. Ramanujam. Semantics of distributed definite clause programs. *Theoretical Computer Science*, 68:203–220, 1989.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, 1986.

[18] Y. Sakakibara. Programming in modal logic: An extension of PROLOG based on modal logic. In *Logic Programming 86*, pages 81–91. Springer LNCS 264, 1986.

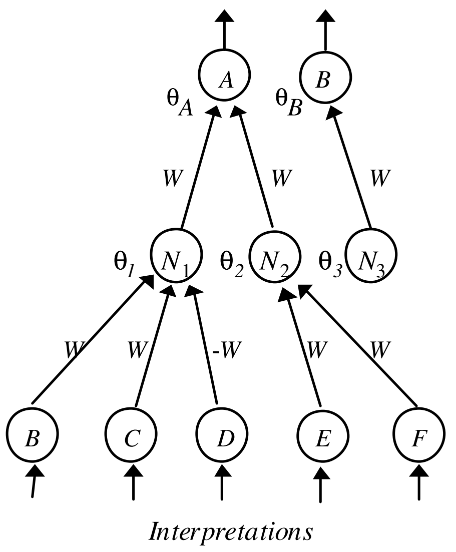

Figure 1: Sketch of a network $\mathcal{N}$ for program $\mathcal{P}$.

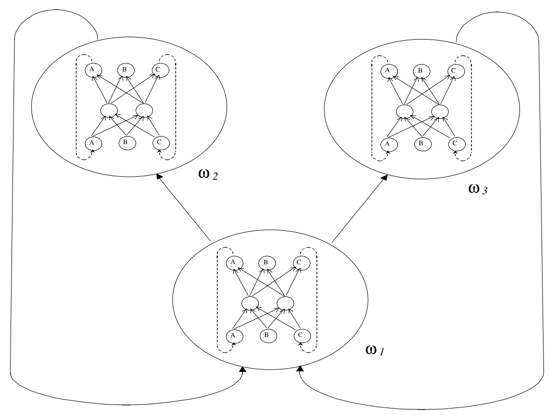

Figure 2: An ensemble of C-IL^2 P networks for modal logic.

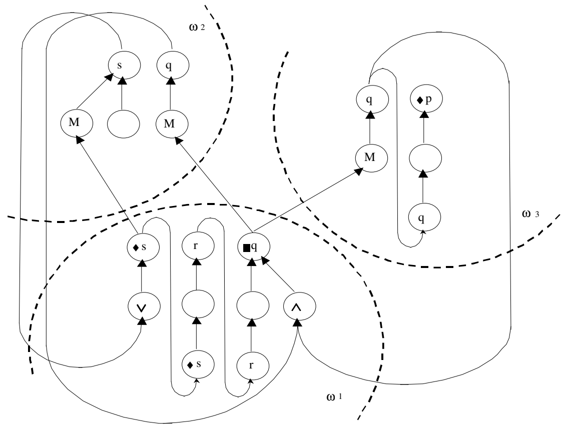

Figure 3: The ensemble $\mathcal{N}_1, \mathcal{N}_2$, $\mathcal{N}_3$ that represents $\mathcal{P}$.