Distributed Knowledge Representation in Neural-Symbolic Learning Systems: A Case Study

Artur S. d'Avila Garcez[δ], Luís C. Lamb[λ], Krysia Broda[β], and Dov M. Gabbay[γ]

[δ]Dept. of Computing, City University, London EC1V 0HB, UK aag@soi.city.ac.uk

[λ]Dept. Informática Teórica, II-UFRGS, Porto Alegre, 91501-970, Brazil lamb@inf.ufrgs.br

[β]Dept. of Computing, Imperial College, London, SW72BZ, UK kb@doc.ic.ac.uk

[γ]Dept. of Computer Science, King's College, London WC2R 2LS, UK dg@dcs.kcl.ac.uk

Abstract

Neural-symbolic integration concerns the integration of symbolic and connectionist systems. Distributed knowledge representation is traditionally seen under a purely symbolic perspective. In this paper, we show how neural networks can represent symbolic distributed knowledge, acting as multi-agent systems with learning capability (a key feature of neural networks). We then apply our approach to the well-known muddy children puzzle, a problem used as a testbed for distributed knowledge representation formalisms. Finally, we sketch a full solution to this problem by extending our approach to deal with knowledge evolution over time.

Introduction

Neural-Symbolic integration concerns the application of problem-specific symbolic knowledge within the neurocomputing paradigm. So far, neural-symbolic systems have not been shown to fully represent and learn more expressive languages such as modal and predicate logics (Cloete & Zurada 2000). In this paper, we investigate the effectiveness of a new connectionist framework for the representation and learning of propositional modal logics by applying it to the well-known muddy children puzzle (Fagin *et al.* 1995). The framework uses Modal Logic Programming (Sakakibara 1986) extended to allow modalities such as necessity and possibility in the head of clauses (Orgun & Ma 1994) as hypothesis language. A *Modalities Algorithm* (d'Avila Garcez, Lamb, & Gabbay 2002) is used to set up an ensemble of *Connectionist Inductive Learning and Logic Programming (C-IL^{2}P)* networks (d'Avila Garcez, Broda, & Gabbay 2002; d'Avila Garcez & Zaverucha 1999), each network being an extension of Holldobler and Kalinke's massively parallel model for Logic Programming (Holldobler, Kalinke, & Storr 1999). The network obtained is an ensemble of simple *C-IL^{2}P* networks, each representing a learnable possible world. As shown in (d'Avila Garcez, Lamb, & Gabbay 2002), the resulting ensemble computes a fixed-point semantics of the original modal theory. As a result, the network ensemble can be seen as a massively parallel system for modal logic programming.

In this paper, we validate the system by applying it to a distributed knowledge representation problem, the muddy children puzzle (Fagin *et al.* 1995). Moreover, we point out the way towards a full solution of the muddy children puzzle by catering for the representation of time in neural networks. Section 2 briefly presents the basic concepts of connectionist modal logic used in this paper. Section 3 describes the Modalities Algorithm that translates extended modal programs into artificial neural networks. In Section 4, we apply the system to the muddy children puzzle and extend the approach in order to consider knowledge evolution through time. In Section 5, we conclude and discuss directions for future work.

Connectionist Modal Logic

Modal logic began with the analysis of concepts such as necessity and possibility under a philosophical perspective (Hughes & Cresswell 1996). A main feature of modal logics is the use of (Kripke) *possible world semantics.* In modal logic, a proposition is necessary in a world if it is true in all worlds which are possible in relation to that world, whereas it is possible in a world if it is true in at least one world which is possible in relation to that same world. This is expressed in the semantics formalisation by a (binary) relation between possible worlds. Modal logic was found to be appropriate to study mathematical necessity (in the logic of provability), time, knowledge and other modalities (Chagrov & Zakharyaschev 1997). In artificial intelligence, modal logics are amongst the most employed formalisms to analyse and represent reasoning in multi-agent systems (Fagin *et al.* 1995). Formally, the language of propositional modal logic extends the language of propositional logic with the $\Box$ and $\Diamond$ operators. Moreover, we assume that any clause is ground over a finite domain (i.e. they contain no variables). Essential definitions are then stated.

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

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

[0]This work was carried out when the second author was an Honourary Visiting Research Fellow at City University London, Fall 2002.

In addition, each clause is labelled by the possible world in which they hold, similarly to Gabbay's Labelled Deductive Systems (Gabbay 1996).

Definition 2 *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.*

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. Formulas in modal logic will be interpreted in Kripke models, i.e. a set of worlds Ω, related by a binary relation $\mathcal{R}$ and an assignment v of worlds to formulas. A modal formula α is said to be true at a possible world ω of a model $\mathcal{M}$, written $(\mathcal{M}, \omega) \models \alpha$, if α holds in ω. α is said to be true at a model $\mathcal{M}$ if it is true in every world in $\mathcal{M}$. The rules we shall represent using *C-IL²P* are similar to the ones presented in (Russo 1996) and are reproduced in Table 1:

Table 1: Rules for modality operators

$\dfrac{[R(\omega, g_\alpha(\omega))]...g_\alpha(\omega) : \alpha}{\omega : \Box\alpha}\ \Box I$	$\dfrac{\omega_1 : \Box\alpha, R(\omega_1, \omega_2)}{\omega_2 : \alpha}\ \Box E$
$\dfrac{\omega : \Diamond\alpha}{f_\alpha(\omega) : \alpha, R(\omega, f_\alpha(\omega))}\ \Diamond E$	$\dfrac{\omega_2 : \alpha, R(\omega_1, \omega_2)}{\omega_1 : \Diamond\alpha}\ \Diamond I$

Semantics for Extended Modal Logic Programs

When computing the semantics of a modal program, we have to consider both the fixed-point of a particular world, and the fixed-point of the program as a whole. When computing the fixed-point in each world, we have to consider the consequences derived locally and the consequences derived from the interaction between worlds. Locally, fixed-points are computed as in the stable model semantics for 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 (Brewka & Eiter 1999) to it. When considering interacting worlds, there are two cases to be addressed, according to the $\Box I$ and $\Diamond I$ rules in Table 1, together with the accessibility relation $\mathcal{R}$, which might render additional consequences in each world. In (d'Avila Garcez, Lamb, & Gabbay 2002) it has been proved that *C-IL²P* ensembles compute a fixed point semantics of the modal theory $\mathcal{P}$ (according to the modal fixpoint operator $MT_\mathcal{P}$ of $\mathcal{P}$), providing the semantics for connectionist modal logic programs.[1]

Theorem 3 *(d'Avila Garcez, Lamb, & Gabbay 2002) 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 fixed-point operator $MT_\mathcal{P}$ of $\mathcal{P}$.*

[1] In this semantics, the choice of an arbitrary world for $\Diamond$ elimination (made before the computation of $MT_\mathcal{P}$) may lead to different fixed-points of a given extended modal program. Such a choice is similar to the approach adopted by Gabbay in (Gabbay 1989) in which one chooses a point in the future for execution and then backtracks if judged necessary (and at all possible).

Any extended modal program ($\mathcal{P}$) can be translated into an ensemble of *C-IL²P* networks ($\mathcal{N}$) with the use of the *Modalities Algorithm* presented below. *C-IL²P* (d'Avila Garcez & Zaverucha 1999) is a massively parallel system based on artificial neural networks that integrates inductive learning from examples and background knowledge with deductive learning from logic programming. Its *Translation Algorithm* maps any general logic program p into a single hidden layer neural network n such that n computes the fixed-point of p. As a generalisation of *C-IL²P*, the *Modalities Algorithm* is used to interconnect the different *C-IL²P* networks n, which will correspond to possible worlds of an extended modal program $\mathcal{P}$, into the ensemble $\mathcal{N}$ that will compute the modal fixed-point of $\mathcal{P}$. By using ensembles of *C-IL²P* networks, we enhance the expressive power of the system, yet maintaining the simplicity needed to perform inductive learning efficiently. More details on the *C-IL²P* system can be found in (d'Avila Garcez, Broda, & Gabbay 2002).

The Modalities Algorithm

1. 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 μ_l, W and θ_A are obtained from *C-IL²P's Translation Algorithm*[2]
2. For each $\mathcal{P}_i$ do: (a) Rename each ML_j in $\mathcal{P}_i$ by a new literal not occurring in $\mathcal{P}$ of the form $L_j^\Box$ if $M = \Box$, or $L_j^\Diamond$ if $M = \Diamond$;[3] (b) Call *C-IL²P's Translation Algorithm;*
3. For each output neuron $L_j^\Diamond$ in $\mathcal{N}_i$, do: (a) Add a hidden neuron L_j^M to an arbitrary $\mathcal{N}_k$ $(0 \leq k \leq n)$ such that $\mathcal{R}(\omega_i, \omega_k)$; (b) Set the step function $s(x)$ 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 $-1 < \theta^M < A_{min}$; (e) Connect L_j^M to L_j in $\mathcal{N}_k$ and set the connection weight to W^M.
4. For each output neuron $L_j^\Box$ in $\mathcal{N}_i$, do: (a) Add a hidden neuron L_j^M to each $\mathcal{N}_k$ $(0 \leq k \leq n)$ such that $\mathcal{R}(\omega_i, \omega_k)$; (b) Set the step function $s(x)$ 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 $-1 < \theta^M < A_{min}$; (e) Connect L_j^M to L_j in $\mathcal{N}_k$ and set the connection weight to W^M.
5. 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 the step function $s(x)$ as the activation function of $L_j^\vee$; (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 $-nA_{min} < \theta^\vee < A_{min} - (n-1)$; (iii) Connect $L_j^\vee$ to $L_j^\Diamond$ in $\mathcal{N}_i$ and set the connection weight to W^M.
6. 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)

[2] μ_l denotes the number of connections to output neuron l. Amin $\in [0, 1]$ denotes the (pre-defined) activation value for a neuron to be considered active (or equivalently for its corresponding literal to be considered true).

[3] This allows us to treat each ML_j as a literal and apply the *Translation Algorithm* directly to $\mathcal{P}_i$, by labelling neurons as $\Box L_j$, $\Diamond L_j$, or L_j.

Set the step function $s(x)$ as the activation function of $L_j^{\wedge}$; (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 $n - (1 + A_{min}) < \theta^{\wedge} < nA_{min}$; (iii) Connect $L_j^{\wedge}$ to $L_j^{\Box}$ in $\mathcal{N}_i$ and set the connection weight to W^M.

Example: 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 1). 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 1.

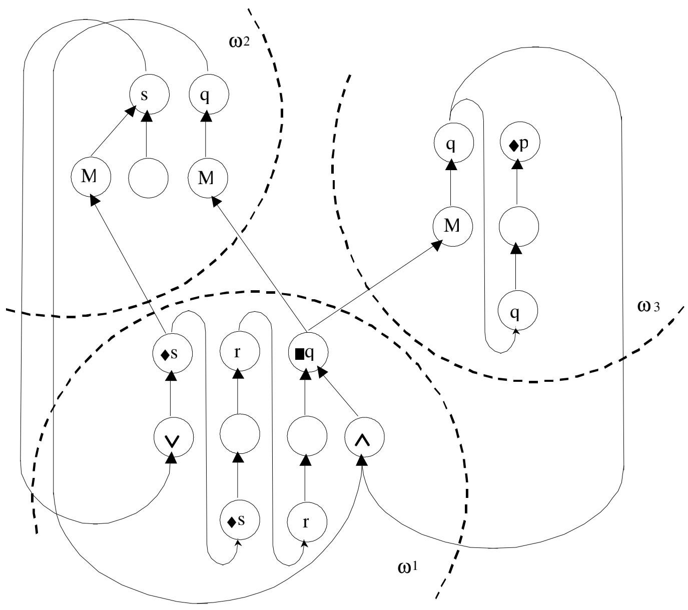

Figure 1: Ensemble $\{\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3\}$ that represents $\mathcal{P}$.

Case Study: Muddy Children Puzzle

In this section, we apply the modal *C-IL²P* system to the muddy children puzzle, a classic example of reasoning in multi-agent environments. The situation in the puzzle is described as follows. *There is a group of n (truthful and intelligent) children playing in a garden. A certain number of children k (k ≤ n) has mud on their faces. Each child can see if the others are muddy, but not herself. Now, consider the following: A caretaker announces that at least one child is muddy* ($k \geq 1$) *and asks "does any of you know if you have mud on your own face?" To help understanding the puzzle, let us consider the cases in which* $k = 1$, $k = 2$ *and* $k = 3$. *If* $k = 1$ *(only one child is muddy), the muddy child answers yes at the first instance since she cannot see any other muddy child. All the other children answer no at the first instance. If* $k = 2$, *suppose children* 1 *and* 2 *are muddy. At first, all children can only answer no. This allows* 1 *to reason as follows: "if* 2 *had said yes the first time, she would have been the only muddy child. Since* 2 *said no, she must be seeing someone else muddy; and since I cannot see anyone else muddy apart from* 2, *I myself must be muddy!" Child* 2 *can reason analogously, and also answers yes the second time round. If* $k = 3$, *suppose children* 1, 2 *and* 3 *are muddy. Every children can only answer no the first two times. Again, this allows* 1 *to reason as follows: "if* 2 *or* 3 *had said yes the second time, they would have been the only two muddy children. Thus, there must be a third person with mud. Since I can see only* 2 *and* 3 *with mud, this third person must be me!" Children* 2 *and* 3 *can reason analogously to conclude as well that yes, they are muddy.*

The above cases clearly illustrate the need to distinguish between an agent's *individual knowledge* and *common knowledge* about the world. For example, when $k = 2$, after everybody says *no* in the first round, it becomes common knowledge that at least two children are muddy. Similarly, when $k = 3$, after everybody says *no* twice, it becomes common knowledge that at least three children are muddy, and so on. In other words, when it is common knowledge that there are at least $k-1$ muddy children; after the announcement that nobody knows if they are muddy or not, then it becomes common knowledge that there are at least k muddy children, for if there were $k - 1$ muddy children all of them would know that they had mud in their faces. Notice that this reasoning process can only start once it is common knowledge that at least one child is muddy, as announced.

Distributed Representation

In our formalisation, a $\mathbf{K}_j$ modality that represents the knowledge of an agent j is used analogously to a $\Box$ modality. In addition, we use p_i to denote that proposition p is *true* for agent i. For example, $\mathbf{K}_j p_i$ means that agent j knows that p is *true* for agent i. We omit the subscript j of $\mathbf{K}$ whenever it is clear from the context. We use p_i to say that child i is muddy, and q_k to say that at least k children are muddy ($k \leq n$). Let us consider the case in which three children are playing in the garden ($n = 3$). Rule r_1^1 below states that when child 1 knows that at least one child is muddy and that neither child 2 nor child 3 are muddy then child 1 knows that she herself is muddy. Similarly, rule r_2^1 states that if child 1 knows that there are at least two muddy children and she knows that child 2 is not muddy then she must also be able to know that she herself is muddy, and so on. The rules for children 2 and 3 are constructed analogously.

Rules for Agent(child) 1:

r_1^1: $\mathbf{K}_1 q_1 \wedge \mathbf{K}_1 \neg p_2 \wedge \mathbf{K}_1 \neg p_3 \rightarrow \mathbf{K}_1 p_1$

r_2^1: $\mathbf{K}_1 q_2 \wedge \mathbf{K}_1 \neg p_2 \rightarrow \mathbf{K}_1 p_1$

r_3^1: $\mathbf{K}_1 q_2 \wedge \mathbf{K}_1 \neg p_3 \rightarrow \mathbf{K}_1 p_1$

r_4^1: $\mathbf{K}_1 q_3 \rightarrow \mathbf{K}_1 p_1$

Each set of rules r_m^l ($1 \leq l \leq n$, $m \in \mathbb{N}^+$) is implemented in a *C-IL²P* network. Figure 2 shows the implementation of rules r_1^1 to r_4^1 (for agent 1)[4]. In addition,

[4]Note that with the use of *classical negation*, $\mathbf{K}p_i$ and $\mathbf{K}\neg p_i$ should be represented as two different input neurons (d'Avila Garcez 2002). Negative weights in the network

it contains p_1[5] and $\mathbf{K}q_1$, $\mathbf{K}q_2$ and $\mathbf{K}q_3$, all represented as facts (highlighted in grey in Figure 2). This setting complies with the presentation of the puzzle given in (Huth & Ryan 2000), in which *snapshots* of the knowledge evolution along time rounds are taken to logically deduce the solution of the problem without the addition of a time variable. In contrast with p_1 and $\mathbf{K}q_k$ ($1 \leq k \leq 3$), $\mathbf{K}\neg\mathrm{p}_2$ and $\mathbf{K}\neg\mathrm{p}_3$ must be obtained from agents 2 and 3, respectively, whenever agent 1 does not see mud on their foreheads.

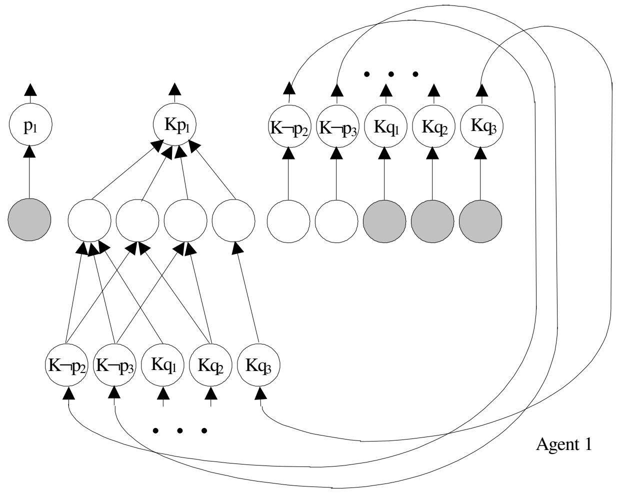

Figure 2: Implementation of rules $\{r_1^1, ..., r_4^1\}$.

Figure 3 illustrates the interaction between three agents in the muddy children puzzle. The arrows connecting *C-IL^{2}P* networks implement the fact that when a child is muddy, the other children can see it. For the sake of clarity, the rules r_m^1, corresponding to neuron $\mathbf{K}_1\mathrm{p}_1$, are shown only in Figure 2. Analogously, the rules r_m^2 and r_m^3 for $\mathbf{K}_2\mathrm{p}_2$ and $\mathbf{K}_3\mathrm{p}_3$ would be represented in similar *C-IL^{2}P* networks. This is indicated in Figure 3 by neurons highlighted in black. In addition, Figure 3 only shows positive information about the problem. Recall that negative information such as $\neg p_1$, $\mathbf{K}\neg p_1$, $\mathbf{K}\neg p_2$ is to be added explicitly to the network, as shown in Figure 2. This completes the translation of a snapshot of the muddy children puzzle in a neural network.

Learning

Experiments we have performed have shown that using the *Modalities Algorithm* to translate a modal background knowledge to the initial ensemble is an effective way of performing learning from examples and background knowledge. We have checked whether particular agents i were able to learn the rules r_m^l. We have run two sets of experiments comparing learning with

---

would then represent *default negation*, allowing one to differentiate between $\mathbf{K}p_i$ and $\sim\mathbf{K}p_i$, and between $\mathbf{K}\neg p_i$ and $\sim\mathbf{K}\neg p_i$, respectively. This can be easily verified by renaming $\mathbf{K}\neg p_i$ by a new literal $\mathbf{K}p_i'$.

[5]Note the difference between p_1 (child 1 is muddy) and $\mathbf{K}p_1$ (child 1 knows that she is muddy).

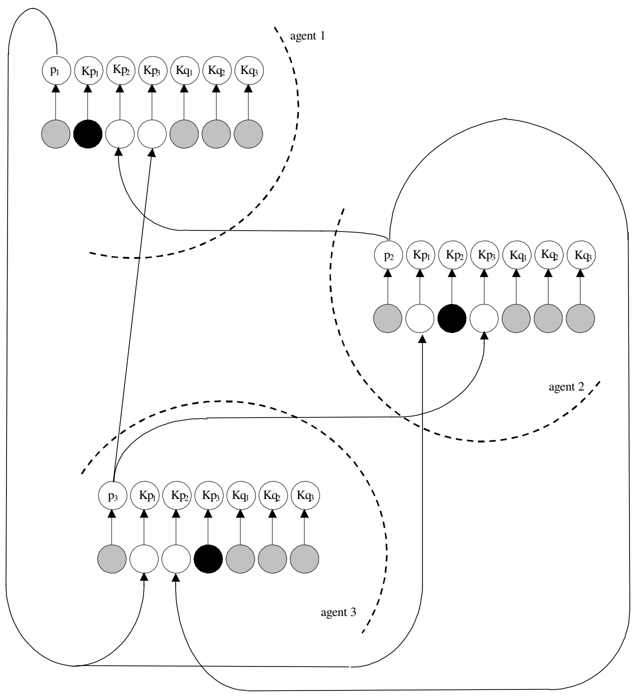

Figure 3: Interaction between agents in the puzzle.

and without background knowledge. Without background knowledge, the networks presented an accuracy of 84.37%, whereas with the addition of the rules r_m^l to the networks, an average accuracy of 93.75% was achieved, corroborating the importance of adding background knowledge.

Towards Temporal Reasoning

The addition of a temporal variable to the muddy children puzzle would allow one to reason about knowledge acquired after each time round. For example, assume as before that three muddy children are playing in the garden. Firstly, they all answer no when asked if they know whether they are muddy or not. Moreover, as each muddy child can see the other children, they will reason as previously described, and answer no the second time round, reaching the correct conclusion in time round three. This solution requires, at each round, that the *C-IL^{2}P* networks be expanded with the knowledge acquired from reasoning about what is seen and what is heard by each agent. This clearly requires each agent to reason about time. The snapshot solution should then be seen as representing the knowledge held by the agents at an arbitrary time t. The knowledge held by the agents at time $t+1$ would then be represented by another set of *C-IL^{2}P* networks, appropriately connected to the original set of networks. This can be visualised in Figure 4 where each dotted box contains the knowledge of a number of agents at a particular time point t_i (such as in Figure 3).

Knowledge evolution over time as presented in Figure 4 allows us to explicitly represent the fact that when it is common knowledge that there are at least $k-1$ muddy children at time t; after the announcement that nobody knows if they are muddy or not, then it becomes common knowledge that there are at least k muddy children at time $t+1$. This is done by interconnecting a number s of network ensembles similar to that depicted in Figure 3. For example, the knowledge of child 1 about the number of muddy children would evolve in time as follows: at time t_1, child 1 knows that there is at least one muddy child ($\mathbf{K}q_1$) since the caretaker had announced so. At time t_2, child 1 will know that there are at least two muddy children ($\mathbf{K}q_2$), provided that no child knew that she was muddy at time t_1 ($\sim\mathbf{K}_i p_i$ for any child i), and so on. It is clear that the knowledge of child 1 is evolving through time until she reaches the conclusion that she helself is muddy at time t_3. Note that at each time transition, child 1 learns whether both children 2 and 3 had answered *no* to the caretaker, allowing her to conclude whether there is yet another muddy child in the garden ($\mathbf{K}q_j$, $1 \leq j \leq 3$).

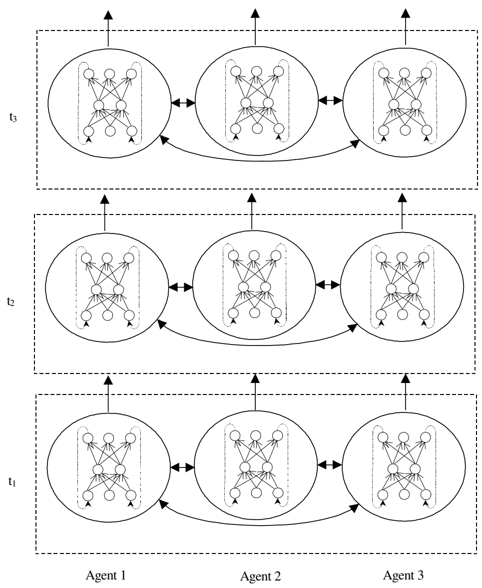

Figure 4: Evolving knowledge through time.

The definition of the number of ensembles s that are necessary to solve a given problem clearly depends on the problem domain, and on the number of time points that are relevant for reasoning about the problem. For example, in the case of the muddy children puzzle, we know that it suffices to have s equals to the number of children that are muddy. The definition of s in a different domain might not be as straightforward, possibly requiring a fine-tuning process similar to that performed during learning but with a varying network architecture.

Conclusion

The connectionist modal logic framework presented here renders Neural-Symbolic Learning Systems with the ability to provide a more expressive representation language. It contributes to the integration of both research programmes - neural networks and modal logics - into a unified foundation. In this paper, we have proposed a solution to the muddy children puzzle where agents can reason about their knowledge of the situation at each time step. In addition, we have seen that the provision of a *Temporal Algorithm*, similar to the *Modalities Algorithm* above, would require knowledge about the problem domain to define the number s of relevant time points. As an alternative, a formalisation of the full solution to the muddy children puzzle would require the addition of a modality to deal with the notion of *next time* in a linear timeflow. This notion of a temporal modality could be implemented in the connectionist modal logic system with the use of both the $\Box$-like modality and the $\Diamond$-like modality. In this case, the network of Figure 4 should be seen as the unfolded version of a recurrent network. We believe that the connectionist modal logic framework investigated here opens several interesting research avenues in the domain of neural-symbolic learning systems, as it allows for the representation and learning of expressive languages of non-classical logics in hybrid systems.

References

Brewka, G., and Eiter, T. 1999. Preferred answer sets for extended logic programs. *AIJ* 109:297–356.

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

Cloete, I., and Zurada, J. M., eds. 2000. *Knowledge-Based Neurocomputing.* The MIT Press.

d'Avila Garcez, A. S., and Zaverucha, G. 1999. The connectionist inductive learning and logic programming system. *Applied Intelligence Journal* 11(1):59–77.

d'Avila Garcez, A. S.; Broda, K.; and Gabbay, D. M. 2002. *Neural-Symbolic Learning Systems: Foundations and Applications.* Springer-Verlag.

d'Avila Garcez, A. S.; Lamb, L. C.; and Gabbay, D. M. 2002. A connectionist inductive learning system for modal logic programming. In *Proc. IEEE ICONIP'02.*

d'Avila Garcez, A. S. 2002. Extended theory refinement in knowledge-based neural networks. In *Proc. IEEE IJCNN'02.*

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

Gabbay, D. M. 1996. *Labelled Deductive Systems.* Clarendom Press.

Gabbay, D. M. 1989. The declarative past and imperative future. LNCS 398, 409–448. Springer-Verlag.

Holldobler, S.; Kalinke, Y.; and Storr, H. P. 1999. Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence Journal* 11(1):45–58.

Hughes, G. E., and Cresswell, M. J. 1996. *A new introduction to modal logic.* Routledge.

Huth, M. R. A., and Ryan, M. D. 2000. *Logic in Computer Science: Modelling and Reasoning about Systems.* Cambridge University Press.

Orgun, M. A., and Ma, W. 1994. An overview of temporal and modal logic programming. LNAI 827, 445–479. Springer.

Russo, A. 1996. Generalising propositional modal logic using labelled deductive systems. In *Frontiers of Combining Systems, (APLS)*, Vol. 3, 57–74. Kluwer.

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