??, ??, 1–16 (??)

Approximating the Semantics of Logic Programs by Recurrent Neural Networks *

STEFFEN HÖLLDOBLER
Artificial Intelligence Institute
Computer Science Department
Dresden University of Technology
D-01062 Dresden, Germany

sh@inf.tu-dresden.de

YVONNE KALINKE**
Neurocomputing Research Center
Queensland University of Technology
Brisbane, Australia, GPO Box 2434, QLD 4001

yvonne@fit.qut.edu.au

HANS–PETER STÖRR
Artificial Intelligence Institute
Computer Science Department
Dresden University of Technology
D-01062 Dresden, Germany

haps@inf.tu-dresden.de

;

Abstract. In [18] we have shown how to construct a 3–layered recurrent neural network that computes the fixed point of the meaning function $T_{\mathcal{P}}$ of a given propositional logic program $\mathcal{P}$, which corresponds to the computation of the semantics of $\mathcal{P}$. In this article we consider the first order case. We define a notion of approximation for interpretations and prove that there exists a 3–layered feed forward neural network that approximates the calculation of $T_{\mathcal{P}}$ for a given first order acyclic logic program $\mathcal{P}$ with an injective level mapping arbitrarily well. Extending the feed forward network by recurrent connections we obtain a recurrent neural network whose iteration approximates the fixed point of $T_{\mathcal{P}}$. This result is proven by taking advantage of the fact that for acyclic logic programs the function $T_{\mathcal{P}}$ is a contraction mapping on a complete metric space defined by the interpretations of the program. Mapping this space to the metric space $\mathbb{R}$ with Euclidean distance, a real valued function $f_{\mathcal{P}}$ can be defined which corresponds to $T_{\mathcal{P}}$ and is continuous as well as a contraction. Consequently it can be approximated by an appropriately chosen class of feed forward neural networks.

Keywords: Recurrent Neural Networks, Logic Programs, Model Generation, Approximations.

1. Introduction

Many researchers are convinced that intelligent agents reason by generating models and deducing conclusions with respect to these models (see e.g. [21]). In real agents the reasoning process itself is performed by highly recurrent neural networks, whose precise structure and functionality is still not very well understood. Artificial neural or connectionist networks are just a rather crude model for such real neural networks. Nevertheless, they serve as a reasonable model and have been taken up by many researchers. Currently one of the major open problems in this area is the question of how the full power of deductive processes can be implemented on connectionist networks (see e.g. [43]). In this article we will focus on this problem by establishing a strong link between model generation in the context of first order logic and recursive artificial neural networks. In particular, we will show that for a certain class of logic programs the least model of a given program can be approximated arbitrarily well by a recursive artificial neural network.

Model generation is a well established area within automated deduction (see e.g. [28, 14, 42]). In particular, the semantics of a logic program $\mathcal{P}$ is often defined with the help of a so–called meaning function $T_\mathcal{P}$. In many but not all cases a logic program $\mathcal{P}$ admits a least model, which can be computed as the least fixed point of $T_\mathcal{P}$, and research is focussed on identifying classes of programs for which such least models exist. For these classes $T_\mathcal{P}$ effectively specifies a model generation procedure. Examples are the class of definite programs (see e.g. [27]), where the correspondence between the least model of $\mathcal{P}$ and the least fixed point of $T_\mathcal{P}$ can be shown by lattice–theoretic arguments [5], or the class of acceptable programs, where the just mentioned correspondence can be shown using metric methods [12].

It turned out that the computation of the least model for a logic program from one of the just mentioned classes can be performed by a recursive network of binary threshold units if the programs are propositional [18]: With each interpretation I a vector of units is identified such that the jth unit is active iff the jth propositional variable is mapped to true by I. Two such vectors serve as input and output layer of a 3–layered feed forward network. The hidden layer is constructed such that it contains a unit for each program clause. Such a unit becomes active as soon as the body of the clause is satisfied by the interpretation represented by the activation pattern of the input layer, and propagates its activation to the unit in the output layer representing the head of the clause. Fig. 1 depicts such a network for a small example. From its construction follows immediately that such a network computes the application of $T_\mathcal{P}$ to an interpretation I. Turning this network into a recursive one by connecting corresponding units from the output to the input layer with weight 1 allows to compute the least fixed point of $T_\mathcal{P}$ for propositional acceptable logic programs.

One should observe that the number of units and the number of connections in a recursive neural network corresponding to a propositional program are bounded by $O(m + n)$ and $O(m \times n)$ respectively, where n is the number of clauses and m is the number of propositional variables occurring in the program. Furthermore, the application of $T_\mathcal{P}$ to a given interpretation is computed in 2 steps. As the sequential time to compute $T_\mathcal{P}$ is bound by $O(n \times m)$ (assuming that no literal occurs more than once in the conditions of a clause), the resulting parallel computational model is optimal.[1]

The approach in [18] provides a new computational model for the computation of the least model of logic programs, which is massively parallel and optimal. Moreover, it establishes a strong relationship between propositional logic programming and recurrent neural networks. In contrast to recurrent neural networks logic programs are well understood. Hence, this new relationship may help to gain a better insight into recursive neural networks, to formally analyze these networks and to give a declarative semantics for what these networks are doing. For example, given an arbitrary recurrent network of the architecture shown in Fig. 1, i.e. a 3–layered feed forward network with binary threshold units, we can extract a propositional logic program $\mathcal{P}$ such that the net-

*This is an extensively revised version of [19]. A brief summary of the main results is contained in [7].

**The author acknowledges support from the German Academic Exchange Service (DAAD) under grant no. D/97/29570.

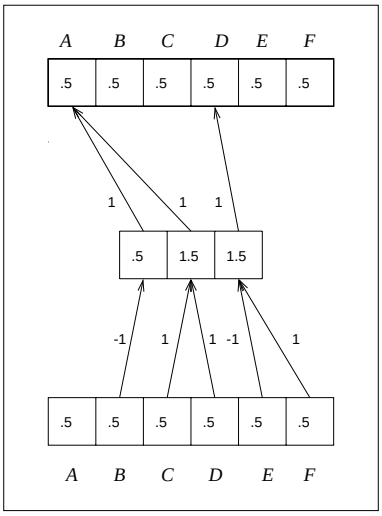

Fig. 1. A feed forward network of binary threshold units computing $T_{\mathcal{P}}$ for the program $\mathcal{P} = \{A \leftarrow \neg B;\ A \leftarrow C, D;\ D \leftarrow \neg E, F\}$. The numbers in the units denote thresholds whereas the numbers at the connections denote weights. Connections with weight 0 are not shown.

work actually computes the meaning function of $\mathcal{P}$. Accordingly the extended recurrent network computes the iteration of the meaning function of $\mathcal{P}$ and, therefore, the semantics of $\mathcal{P}$.

In [11] it was shown that all results of [18] continue to hold if the threshold units of the hidden layer are replaced by sigmoidal ones and the weights on the connections are replaced by real numbers. With this replacement powerful learning techniques like backpropagation can be applied to adapt an initially given logic program. For example, we can now train the feed forward network corresponding to the program with knowledge not yet included in the program and given in form of data examples. Thereafter, a (hopefully) improved logic program can be extracted from the trained network using well–known rule extraction techniques (see e.g. [11, 46, 2]).

All results mentioned so far are propositional in nature. The logic programs are propositional ones, but also the rule extraction techniques mentioned in the previous paragraph generate only propositional rules. In fact, most of the research done so far and aiming at using neural networks for inferential processes is limited to propositional inference systems. This has already been observed by McCarthy in 1988 [29] and not much has changed since. For example, Pinkas has shown that finding a global minima in a Hopfield network corresponds to finding a model of a propositional logic formula and vice versa [33]. He also showed that Hopfield networks can be used to compute the models of first order formulas if these formulas or parts thereof may not be copied [32]. Because it cannot be determined in advance how many copies of a formula or parts thereof are needed in a proof of a first order formula, a system may fail to find a proof if the number of copies is restricted. Without the ability to copy formulas or parts thereof the logic is effectively propositional. This holds for Pinkas' logic but also for SHRUTI [40] and CHCL [17], to mention just a few.

There is an additional problem related to the complexity of objects that can be handled by connectionist inference systems. Systems like SHRUTI [40] and ROBIN [26] allow only constants. They cannot handle structured objects at all. Systems like CHCL [17] and the one by Pinkas [32] allow first order terms. But because they do not allow to copy formulas or parts thereof, their ability to generate new terms is limited. Other proposals like the recursive auto–associative memory [36], its labeled version [44] and holographic reduced representations [34] allow to represent structured objects in principle, but extensive tests revealed that these representations become extremely noisy and effectively useless as soon as more or less complex terms are stored [22].

Investigations of the computational capabilities of recursive neural network have led to promising results about their ability to deal with the more complex class of first order logic programs. In [41] it has been proven that a recursive neural network made up of neurons using linear saturation activation functions can simulate an universal Turing machine. Other approaches relate recursive neural networks to deterministic finite state machines (for first order networks see [31, 8, 1]; for second order networks see [30, 47, 16]; for radial basis function networks see [13]); for iterated function systems see [24, 23]). But as shown for varying examples (see e.g. [45, 25]) constraints on the network architecture may reduce the computational power of the considered network and the assessment of the computational power of a specific architecture doubtless is of great importance.

Summing up, we are unaware of a single connectionist inference system, whose ability to handle structured objects and structure–sensitive processes comes even close to the degree that is achieved in conventional, first order inference systems. Consequently, if structured objects and structure–sensitive processes are to be modeled in connectionist systems, we should take into account the vast knowledge accumulated in conventional systems so far. One step in this direction and the step taken herein is to design a truly connectionist model for a first order inference system. On the other hand, connectionist systems exhibit many desirable properties like fault tolerance and graceful degradation which conventional systems are lacking. If these properties are to be preserved in a connectionist inference system, then very likely we have to give up correctness and/or completeness of the inference system. So we set ourself the goal to approximate the (correct and complete) models of a first order formula.

More precisely, in this article we deal with the question whether the neural architecture used in the case of propositional logic programs [18] is capable and sufficient for the first order case as well. Is a 3–layered feed forward network capable of computing the meaning function for a first order logic program $\mathcal{P}$? Is the corresponding recursive network capable of computing or approximating the iteration of the meaning function? What precisely is the approximation of a meaning function? What precisely is the approximation of the semantics of a logic program? Can we establish a relationship between first order logic programs and 3–layered recursive networks? How do such networks look like?

Because in the first order case an interpretation may be an infinite subset of the set of ground atoms with respect to a given alphabet, the construction developed in [18] may lead to infinite networks if directly applied to first order logic programs. This is unacceptable in practice, where we can handle only finite networks. On the other hand, we may use real valued units with sigmoidal activation functions instead of the binary threshold units used in the propositional case.

Feed forward networks with at least one hidden layer of real valued units with sigmoidal activation function are known to approximate continuous real valued functions as well as Borel measurable functions arbitrarily well [15, 20]. This gives rise to the following idea: Can we find a real valued function $f_{\mathcal{P}}$ corresponding to $T_{\mathcal{P}}$ such that these results can be used to approximate $f_{\mathcal{P}}$ and, thereby, $T_{\mathcal{P}}$ arbitrarily well? If such a function $f_{\mathcal{P}}$ exists and we find a feed forward network approximating $f_{\mathcal{P}}$ to a desired degree of accuracy, can we then turn this network into a recurrent one, such that the recurrent network approximates the least fixed point of $T_{\mathcal{P}}$ arbitrarily well?

In this article we define a class of first order logic programs such that both questions are answered positively for this class. We also discuss what precisely is denoted by an approximation of the least model of a first order logic program. For the first time this gives us a strong link between first order model generation and the computation performed by a recursive neural network.

The remainder of this article is organized as follows. After stating some preliminaries concerning logic programs and metric spaces in Section 2, we state in Section 3 that the meaning function $T_{\mathcal{P}}$ for a recurrent logic program has a unique fixed point that can be computed by iterating $T_{\mathcal{P}}$. In Section 4 we show how to encode the domain of the meaning function $T_{\mathcal{P}}$ into $\mathbb{R}$. In Section 5 we use this mapping to construct a real valued function $f_{\mathcal{P}}$ corresponding to the function $T_{\mathcal{P}}$. In Section 6 we show that the real valued function $f_{\mathcal{P}}$ is continuous for a certain class of programs and that a feed forward network with sigmoidal activation functions and at least one hidden layer can approximate the function $f_{\mathcal{P}}$. Thereafter we show how to extend the feed forward network to a recurrent one such that the iteration of $f_{\mathcal{P}}$ corresponds to the iteration of $T_{\mathcal{P}}$ and, in fact, approximates the least fixed point of $T_{\mathcal{P}}$. Because in the case of recurrent programs such a fixed point does always exist we end up with approximating the semantics of the logic program $\mathcal{P}$. Finally, we discuss our results and point out future work in Section 9. The formal proofs of our results can be found in the appendix of the article.

2. Preliminaries

In the following two subsections we briefly recall basic notions and notations concerning logic

programs according to [27] and metric spaces as in [12].

2.1. Logic Programs

A *logic program* $\mathcal{P}$ is a collection of (universally closed) clauses of the form $A \leftarrow L_1, \ldots, L_n$, where $n \geq 0$, A is a first order atom and L_i, $1 \leq i \leq n$, are first order literals, i.e. atoms or negated atoms; A is called *head* and $L_1, \ldots, L_n$ *body* of a clause. Each program is assumed to be based on an underlying alphabet with a finite number of function and relation symbols and a countably infinite number of variables. An atom, literal, clause or program is said to be *ground* if it does not contain an occurrence of a variable. A program $\mathcal{P}$ is called *definite* if each clause occurring in $\mathcal{P}$ contains only atoms. $B_\mathcal{P}$ denotes the *Herbrand base*, i.e. the set of ground atoms, with respect to the alphabet underlying $\mathcal{P}$. A *level mapping* for a program $\mathcal{P}$ is a function $|\ | : B_\mathcal{P} \rightarrow \mathbb{N} \setminus \{0\}$, where $\mathbb{N}$ denotes the set of natural numbers. If $|A| = n$ we will say the *level* of the atom $A \in B_\mathcal{P}$ is n. A level mapping is extended to literals by defining $|\neg A| = |A|$.

An *interpretation* is a mapping from ground atoms to the set $\{\mathbf{1}, \mathbf{0}\}$ of truth values. As usual we represent interpretations by the set $I \subseteq B_\mathcal{P}$ of atoms mapped to $\mathbf{1}$; using such a representation it is convenient to agree that $\neg A \in I$ iff $A \notin I$. Interpretations are extended to literals, clauses and programs in the standard way. A *model* for $\mathcal{P}$ is an interpretation which maps $\mathcal{P}$ to $\mathbf{1}$. The *meaning function* $T_\mathcal{P} : 2^{B_\mathcal{P}} \rightarrow 2^{B_\mathcal{P}}$ is defined as follows: Let I be an interpretation and A a ground atom. $A \in T_\mathcal{P}(I)$ iff there exists a ground instance $A \leftarrow L_1, \ldots, L_n$ of a clause in $\mathcal{P}$ such that for all $1 \leq i \leq n$ we find $L_i \in I$.

2.2. Metric Spaces

A *metric* or *distance function* on a space $\mathcal{M}$ is a mapping $d : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}^{\geq 0}$ such that $d(x, y) = 0$ iff $x = y$, $d(x, y) = d(y, x)$, and $d(x, y) \leq d(x, z) + d(z, y)$, where $\mathbb{R}^{\geq 0}$ denotes the set of non–negative real numbers. Let $(\mathcal{M}, d)$ be a metric space and $\mathcal{S} = s_1,\ s_2,\ \ldots,\ s_i \in \mathcal{M}$, be a sequence on $\mathcal{M}$. $\mathcal{S}$ *converges* if $\exists s \in \mathcal{M} : \forall \varepsilon > 0 : \exists N : \forall n \geq N : d(s_n, s) \leq \varepsilon$. $\mathcal{S}$ is *Cauchy* if $\forall \varepsilon > 0 : \exists N : \forall n, m \geq N : d(s_n, s_m) \leq \varepsilon$. $(\mathcal{M}, d)$ is *complete* if every Cauchy sequence converges. A mapping $f : \mathcal{M} \rightarrow \mathcal{M}$ is a *contraction* on $(\mathcal{M}, d)$ if $\exists 0 < k < 1 : \forall x, y \in \mathcal{M} : d(f(x), f(y)) \leq k \cdot d(x, y)$.

Consider a logic program $\mathcal{P}$, a level mapping $|\ |$ for $\mathcal{P}$ and the set $2^{B_\mathcal{P}}$ of interpretations for $\mathcal{P}$. A distance function $d_\mathcal{P}$ on $2^{B_\mathcal{P}}$ associated with $|\ |$ is defined as follows. Let I and J be two interpretations. If $I = J$ then $d_\mathcal{P}(I, J) = 0$. Otherwise, $d_\mathcal{P}(I, J) = 2^{-n}$, where I and J differ on some atom A of level n but agree on all atoms of lower level. As shown in [12] the distance function $d_\mathcal{P}$ associated with a level mapping $|\ |$ for $\mathcal{P}$ is a metric and the metric space $(2^{B_\mathcal{P}}, d_\mathcal{P})$ is complete.

For complete metric spaces and contraction mappings the following theorem is well–known:

Theorem 1. (Banach Contraction Theorem [48]) *A contraction mapping f on a complete metric space has a unique fixed point. The sequence $x,\ f(x),\ f(f(x)), \ldots$ converges to this fixed point for any x.*

3. Acyclic Logic Programs

Acyclic logic programs were investigated by Cavedon [9][2] as well as Apt and Bezem [3, 4]. They are a subclass of the class of locally stratified programs defined by Przymusinsky [37] and enjoy several desirable properties like the fact that for each acyclic program $\mathcal{P}$ the meaning function $T_\mathcal{P}$ has a unique fixed point $M_\mathcal{P}$, that $M_\mathcal{P}$ is a minimal model and the perfect model[3] of $\mathcal{P}$, and that $M_\mathcal{P}$ is the unique Herbrand model of the completion of $\mathcal{P}$.[4] In this paper we will show that acyclic logic programs exhibit an additional characteristic that allows to approximate their semantics by a recursive neural network. This is possible because by using metric methods as proposed by Fitting [12] we can show that the meaning function $T_\mathcal{P}$ for an acyclic logic program $\mathcal{P}$ is a contraction on an appropriately defined complete metric space.

Definition 1. A logic program is called *acyclic with respect to a level mapping* $|\ |$, if for every ground instance $A \leftarrow L_1, \ldots, L_n$ of a clause in $\mathcal{P}$:

$$\forall 1 \leq i \leq n :\ |A| > |L_i|.$$

$\mathcal{P}$ is called *acyclic* if $\mathcal{P}$ is acyclic with respect to some level mapping.

Proposition 1. *Let $\mathcal{P}$ be an acyclic logic program with respect to a level mapping $|\ |$ and $d_{\mathcal{P}}$ the distance function associated with the level mapping $|\ |$. The meaning function $T_{\mathcal{P}}$ is a contraction on the complete metric space $(2^{B_{\mathcal{P}}}, d_{\mathcal{P}})$.*

The formal proof of this result and all other proofs are given in the appendix. This result was independently achieved by Seda and Hitzler in [38], who showed that for an even larger class of programs the meaning function is a contraction. Seda and Hitzler are interested in relating metric methods and, in particular, the theory of dynamical systems to logic programs (see also [39]). For the purpose of this paper, viz. to relate logic programs and recursive neural networks, we concentrate on acyclic programs.

Applying Theorem 1 to Proposition 1 we find that for acyclic logic programs $\mathcal{P}$ the meaning function $T_{\mathcal{P}}$ has a unique fixed point and, moreover, this fixed point is reached by iterating $T_{\mathcal{P}}$ starting from any interpretation $I \in B_{\mathcal{P}}$.

4. Mapping Interpretations to Real Numbers

Since we are interested in first order logic programs, the arguments of the meaning function $T_{\mathcal{P}}$ are interpretations which may consist of a countably infinite number of ground atoms. As already mentioned in the introduction the simple solution for the propositional case as presented in [18], where each ground atom is represented by a binary threshold unit in the input and output layer is no longer feasible. To extend the representational capability of our network we use real valued units with sigmoidal activation functions instead. Thus interpretations are to be represented by real numbers.

In this section we define an encoding R of the domain $2^{B_{\mathcal{P}}}$ of the meaning function $T_{\mathcal{P}}$ into $\mathbb{R}$. Considering our aim of approximating $T_{\mathcal{P}}$ by the use of a feed forward network, we should make sure that the encoding of $T_{\mathcal{P}}$ in terms of a real valued function $f_{\mathcal{P}}$ is done in a way such that $f_{\mathcal{P}}$ can indeed be approximated. For example, if $f_{\mathcal{P}}$ would be continuous, then we could apply the result of [15] that continuous functions can be approximated arbitrarily well by a feed forward network.[5] As we shall see, by restricting ourselves to a certain class of programs we can ensure that the function $f_{\mathcal{P}}$ encoding $T_{\mathcal{P}}$ is continuous.

We start from a level mapping $|\ | : B_{\mathcal{P}} \to \mathbb{N}$ from ground atoms to natural numbers. We further require it to be injective, and express this by renaming it to $\|\ \|$. This restriction is discussed further in Section 7. It is straightforward to generate an injective level mapping for a given program because the set of ground atoms over an alphabet with a finite number of function and predicate symbols can be enumerated. The property of being an acyclic program is undecidable (see [3]) and we conjecture that it is also undecidable whether a given program is acyclic with respect to an injective level mapping.

We use the level mapping $\|\ \|$ to define a mapping R from the set $2^{B_{\mathcal{P}}}$ of interpretations for a logic program $\mathcal{P}$ to real numbers:

Definition 2. The mapping $R : 2^{B_{\mathcal{P}}} \to \mathbb{R}$ is defined as $R(I) = \sum_{A \in I} 4^{-\|A\|}$. The set $\mathcal{D}_f = \{r \in \mathbb{R} \mid \exists I \in 2^{B_{\mathcal{P}}} :\ r = R(I)\}$ is the *range* of R.

Thus an interpretation I is mapped to the real number whose n–th digit after the comma in the quaternary[6] number system is 1 if there is an atom A in I with $\|A\| = n$,[7] and 0 otherwise. Observe that R is monotone, $R(\emptyset) = 0$ and $R(2^{B_{\mathcal{P}}}) = \frac{1}{3}$.[8] Consequently, the range $\mathcal{D}_f$ of R is a subset of $[0, \frac{1}{3}]$.

Proposition 2. *The set $\mathcal{D}_f$ is a closed subset of $\mathbb{R}$.*

Consequently, $\mathcal{D}_f$ with the Euclidean distance is a complete metric space as well. To proof this proposition we make use of a close relation between the metric $d_{\mathcal{P}}$ on interpretations and the Euclidean distance of the corresponding real numbers:

Proposition 3. *For two interpretations $I, J \in 2^{B_{\mathcal{P}}}$ the following holds:*

$$\frac{2}{3} d_{\mathcal{P}}(I,J)^2 \leq |R(I) - R(J)| \leq \frac{4}{3} d_{\mathcal{P}}(I,J)^2 \ .$$

The requirement of $\| \ \|$ to be injective ensures that R is injective and thus there is an inverse mapping R^{-1}. This inverse mapping is extended to a function $R^{-1} : \mathbb{R} \to 2^{B_\mathcal{P}}$ by taking the value I of the nearest point $R(I)$ for $I \in 2^{B_\mathcal{P}}$. Such a point exists, because by Proposition 2 the set $\mathcal{D}_f$ is closed. If there are two such points, we take the lower one.

5. Mapping $T_\mathcal{P}$ to a Real Valued Function $f_\mathcal{P}$

The encoding R maps the elements of the domain of $T_\mathcal{P}$ to real numbers. Hence, we immediately obtain a function $f_\mathcal{P}$ on $\mathbb{R}$ such that the application of $f_\mathcal{P}$ to a real number $r = R(I)$ encoding the interpretation I is equivalent to applying $T_\mathcal{P}$ to $I = R^{-1}(r)$. Figure 2 depicts the relation between $T_\mathcal{P}$ and $f_\mathcal{P}$. One should observe that the encoding R defined in Definition 2 is just an example for mapping $2^{B_\mathcal{P}}$ to $\mathbb{R}$. We can use any mapping which transfers the contraction property of $T_\mathcal{P}$ to $f_\mathcal{P}$, i.e. for which $f_\mathcal{P}$ is a contraction on the real numbers with the Euclidean distance iff $T_\mathcal{P}$ is a contraction on $(2^{B_\mathcal{P}}, d_\mathcal{P})$.

Definition 3. Let $\mathcal{P}$ be a logic program, $B_\mathcal{P}$ its Herbrand base and $T_\mathcal{P}$ the meaning function associated with $\mathcal{P}$. Let R be an encoding of $2^{B_\mathcal{P}}$ in $\mathbb{R}$ and the closed set $\mathcal{D}_f \subseteq \mathbb{R}$ the range of R. The real valued function $\bar{f}_\mathcal{P}$ corresponding to $T_\mathcal{P}$ is defined by

$$\bar{f}_\mathcal{P} : \mathcal{D}_f \to \mathcal{D}_f : r \mapsto R(T_\mathcal{P}(R^{-1}(r))).$$

The function $\bar{f}_\mathcal{P}$ is extended to a function $f_\mathcal{P} : \mathbb{R} \to \mathbb{R}$ by linear interpolation:

$$f_\mathcal{P}(r) = \begin{cases} \bar{f}_\mathcal{P}(r) & \text{if} \quad r \in \mathcal{D}_f \\ \bar{f}_\mathcal{P}(\min(\mathcal{D}_f)) & \text{if} \quad r < \min(\mathcal{D}_f) \\ \bar{f}_\mathcal{P}(\max(\mathcal{D}_f)) & \text{if} \quad r > \max(\mathcal{D}_f) \\ \frac{(M_r - r)}{M_r - m_r} \bar{f}_\mathcal{P}(m_r) + \frac{(r - m_r)}{M_r - m_r} \bar{f}_\mathcal{P}(M_r) & \\ & \text{otherwise,} \end{cases}$$

where $m_r = \max(\mathcal{D}_f \cap [0, r])$ and $M_r = \min(\mathcal{D}_f \cap [r, 1])$ are the greatest point of $\mathcal{D}_f$ below r and least point of $\mathcal{D}_f$ above r respectively.

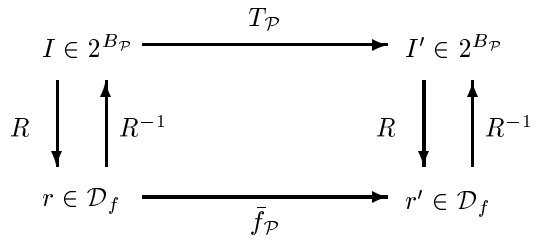

Fig. 2. The relation between $T_\mathcal{P}$ and $\bar{f}_\mathcal{P}$.

One should observe that m_r and M_r are well–defined because $\mathcal{D}_f$ is a closed set by Proposition 2 and $\mathcal{D}_f \cap [0, r]$ as well as $\mathcal{D}_f \cap [r, 1]$ are intersections of closed sets and, thus, are also closed.

In order to show that $f_\mathcal{P}$ is continuous we first prove the following property of $f_\mathcal{P}$:

Proposition 4. *Let $\mathcal{P}$ be an acyclic logic program with an injective level mapping, $T_\mathcal{P}$ the meaning function associated with $\mathcal{P}$ and $f_\mathcal{P}$ the real valued function corresponding to $T_\mathcal{P}$. Then $f_\mathcal{P}$ is a contraction on $\mathbb{R}$, i.e.*

$$\forall r, r' \in \mathbb{R} : |f_\mathcal{P}(r) - f_\mathcal{P}(r')| \leq \frac{1}{2} |r - r'| \ .$$

As an immediate consequence of Proposition 4 we obtain:

Corollary 1. *Let $\mathcal{P}$ be an acyclic logic program with an injective level mapping, $T_\mathcal{P}$ the meaning function associated with $\mathcal{P}$ and $f_\mathcal{P}$ the real valued function corresponding to $T_\mathcal{P}$. Then $f_\mathcal{P}$ is continuous.*

6. Approximating the Meaning Function $T_\mathcal{P}$

Because $f_\mathcal{P}$ is a continuous real valued function we can apply the following theorem stating the approximation capability of a class of feed forward networks.

Theorem 2. *[15] Let $\phi(x)$ be a non constant, bounded and monotone increasing continuous function. Let K be a compact subset (bounded closed subset) of $\mathbb{R}^n$ and $f(x_1, \ldots, x_n)$ be a continuous real valued function on K. Then for an arbitrary $\varepsilon > 0$, there exist an integer N and*

real constants c_i, θ_i $(i = 1, \ldots, N)$, $w_{ij} (i = 1, \ldots, N, j = 1, \ldots, n)$ *such that*

$$\tilde{f}(x_1, \ldots, x_n) = \sum_{i=1}^{N} c_i \phi \left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i \right)$$

satisfies

$$\max_{\bar{x} \in K} |f(x_1, \ldots, x_n) - \tilde{f}(x_1, \ldots, x_n)| < \varepsilon \ .$$

In other words, for an arbitrary $\varepsilon > 0$, there exists a 3–layered feed forward network whose output function for the hidden layer is $\phi(x)$, its output functions for input and output layers are linear and it has an input–output function $\tilde{f}(x_1, \ldots, x_n)$ such that

$$\max_{\bar{x} \in K} |f(x_1, \ldots, x_n) - \tilde{f}(x_1, \ldots, x_n)| < \varepsilon \ .$$

Applying Theorem 2 to Corollary 1, which states that $f_{\mathcal{P}}$ is continuous, we obtain the following theorem:

Theorem 3. *Let $\mathcal{P}$ be an acyclic logic program with an injective level mapping, $T_{\mathcal{P}}$ the meaning function associated with $\mathcal{P}$ and $f_{\mathcal{P}}$ the continuous real valued function corresponding to $T_{\mathcal{P}}$. Then for an arbitrary $\varepsilon > 0$, there exists a feed forward network with sigmoidal activation function for the hidden layer units and linear activation functions for the input and output layer units computing the function $\tilde{f}_{\mathcal{P}}$ which satisfies*

$$\max_{x \in [-1,1]} |f_{\mathcal{P}}(x) - \tilde{f}_{\mathcal{P}}(x)| < \varepsilon \ .$$

This theorem states that given a certain accuracy we can construct a feed forward network that approximates $f_{\mathcal{P}}$ to this desired degree of accuracy. Using the mapping R^{-1} we thereby obtain a feed forward network capable of approximating the meaning function $T_{\mathcal{P}}$ for an acyclic logic program $\mathcal{P}$ with injective level mapping. But what precisely is the approximation of such a meaning function in terms of interpretations? To answer this question we first present an example and, thereafter, a formal definition in the following section.

7. Approximation of Interpretations

Consider the program

$$\begin{aligned} \mathcal{P} : \ & p(0), \\ & p(s(X)) \ \leftarrow \ p(X), \end{aligned}$$

where X is a variable, 0 a constant, s an unary function symbol and p an unary predicate symbol. The level mapping

$$\|p(s^n(0))\| = n + 1$$

for $n \in \mathbb{N}$ is injective, where $s^0(0) = 0$ and $s^{n+1}(0) = s(s^n(0))$. The program $\mathcal{P}$ is acyclic with respect to this level mapping $\| \ \|$. In this case we find

$$\begin{aligned} f_{\mathcal{P}}(R(I)) &= 4^{-\|p(0)\|} + \textstyle\sum_{p(X) \in I} 4^{-\|p(s(X))\|} \\ &= 4^{-\|p(0)\|} + \textstyle\sum_{p(X) \in I} 4^{-(\|p(X)\|+1)} \\ &= \tfrac{1}{4} + \tfrac{1}{4} R(I) \ . \end{aligned}$$

One should observe that $B_{\mathcal{P}}$ is the least model of this program and, by the definition of R,

$$R(B_{\mathcal{P}}) = \frac{1}{3},$$

which also happens to be the least fixed point of $f_{\mathcal{P}}$.

The iteration of $T_{\mathcal{P}}$, which yields the semantics of the program $\mathcal{P}$, is approximated by the iteration of $\tilde{f}_{\mathcal{P}}$. What happens during this iteration? If we approximate $f_{\mathcal{P}}$ to an accuracy ε the evaluation of the approximation $\tilde{f}_{\mathcal{P}}$ yields a value

$$\tilde{f}_{\mathcal{P}}(x) \in [\frac{1+x}{4} - \varepsilon, \frac{1+x}{4} + \varepsilon].$$

Thus, if x is in the interval $[a, b]$, the result $\tilde{f}_{\mathcal{P}}(x)$ is in the interval

$$[\frac{a - \frac{1-4\varepsilon}{3}}{4} + \frac{1-4\varepsilon}{3}, \frac{b - \frac{1+4\varepsilon}{3}}{4} + \frac{1+4\varepsilon}{3}].$$

Applying this argument again one can observe that the next iteration will yield a value $\tilde{f}^2_{\mathcal{P}}(x)$ within the interval

$$[\frac{a-\frac{1-4\varepsilon}{3}}{4^2}+\frac{1-4\varepsilon}{3},\frac{b-\frac{1+4\varepsilon}{3}}{4^2}+\frac{1+4\varepsilon}{3}],$$

and the k-th iteration $\tilde{f}^k_{\mathcal{P}}(x)$ will shrink the interval to

$$[\frac{a-\frac{1-4\varepsilon}{3}}{4^k}+\frac{1-4\varepsilon}{3},\frac{b-\frac{1+4\varepsilon}{3}}{4^k}+\frac{1+4\varepsilon}{3}].$$

It is easy to see that in the limit the iteration of $\tilde{f}_{\mathcal{P}}$ yields a value within

$$[\frac{1-4\varepsilon}{3},\frac{1+4\varepsilon}{3}],$$

although it does not necessarily converge to a fixed value within this interval. If we convert such a value

$$r\in[\frac{1-4\varepsilon}{3},\frac{1+4\varepsilon}{3}]$$

with R^{-1} back to $B_{\mathcal{P}}$ we see that

$$p(s^n(0))\in R^{-1}(r)$$

for $n<-\log_4\varepsilon-1$, which coincides with the least model

$$\{p(0),p(s(0)),p(s(s(0))),\ldots\}$$

of $\mathcal{P}$. The values for $p(s^n(0))$ with $n\geq-\log_4\varepsilon-1$ may differ from the intended model.

This example clarifies our notion of an approximation of an interpretation or a model:

Definition 4. Let $\mathcal{P}$ be a logic program and $|\ |$ a level mapping for $\mathcal{P}$. An interpretation I *approximates* an interpretation J to a degree $N\in\mathbb{N}$ with respect to the level mapping $|\ |$, if for all atoms $A\in B_{\mathcal{P}}$ with $|A|<N$, $A\in I$ iff $A\in J$.

Equivalently, I approximates J to a degree N iff $d_{\mathcal{P}}(I,J)\leq 2^{-N}$. One should observe that in our example we are not only able to approximate the results of the calculation of $T_{\mathcal{P}}$ but are also able to approximate the least fixed point of $T_{\mathcal{P}}$.

Unfortunately, not all acyclic logic programs admit an injective level mapping. Consider for instance the logic program

$$\mathcal{Q}: q(a)\leftarrow p(f(X)),$$

where X is a variable, a a constant, f an unary function symbol and p as well as q are unary predicate symbols. It is acyclic with respect to the level mapping, which maps each ground instance of $p(f(X))$ to 1 and $q(a)$ to 2. But it does not admit an injective level mapping for which $T_{\mathcal{Q}}$ is a contraction because the level of the atom $q(a)$ has to be greater than the maximum of the level of all ground instances of $p(f(X))$, which are infinitely many. So the restriction to acyclic programs that admit an injective level mapping does not allow to treat programs that contain clauses with variables occurring in the body but not in the head of the clause.

Note, moreover, that the class of acyclic logic programs contains not only definite programs but negation is allowed and we can handle acyclic programs with negation as well.

8. Iteration of the Approximation $\tilde{f}_{\mathcal{P}}$

So far we have shown how to approximate the meaning function $T_{\mathcal{P}}$. However, we are mainly interested in the approximation of the least fixed point of $T_{\mathcal{P}}$. From Theorem 3 we have learned that there exits a feed forward network computing $\tilde{f}_{\mathcal{P}}$, which is the approximation of $f_{\mathcal{P}}$, which in turn represents $T_{\mathcal{P}}$. Given such a feed forward network we can turn it into a recursive network by adding recurrent connections with weight 1 between corresponding units in the output and input layer of the feed forward network. Does such a recursive network approximate the least fixed point of $T_{\mathcal{P}}$? Whereas Theorem 3 tells us the activation value $\tilde{f}_{\mathcal{P}}(r)$ is within ε of $f_{\mathcal{P}}(r)$ after the first iteration of the recursive network, this small difference could lead to a bigger difference in each step. Our final theorem tells us that this is not the case with the chosen encoding R:

Theorem 4. *Let $\mathcal{P}$ be an acyclic logic program with an injective level mapping, $T_{\mathcal{P}}$ the meaning function associated with $\mathcal{P}$ and $M_{\mathcal{P}}$ the least fixed point of $T_{\mathcal{P}}$. For an arbitrary $N\in\mathbb{N}$ there exists a recursive network with sigmoidal activation function for the hidden layer units and linear activation functions for the input and output layer*

units computing a function $\tilde{f}_{\mathcal{P}}$ such that there exists an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and for all $x \in [-1, 1]$ the relation

$$d_{\mathcal{P}}(R^{-1}(\tilde{f}^n_{\mathcal{P}}(x)), M_{\mathcal{P}}) \leq 2^{-N}$$

holds.

This theorem tells us that for any initial value $x \in [-1, 1]$ the interpretation represented by $\tilde{f}^n_{\mathcal{P}}(x)$, i.e. the n–fold iteration of the recurrent network, and the least fixed point $M_{\mathcal{P}}$ of $T_{\mathcal{P}}$ agree in all atoms with a level less than N. In other words, we can approximate the least fixed point of the meaning function $T_{\mathcal{P}}$ of an acyclic logic program $\mathcal{P}$ with an injective level mapping arbitrarily well with a recurrent neural network.

9. Discussion and Future Work

In [18] we have shown that for each propositional logic program $\mathcal{P}$ there exists a recurrent neural network computing the semantics of $\mathcal{P}$. In this article we have extended this result to a class of first order logic programs. Because of the infinity of interpretations in the first order case we formally introduced approximations of interpretations and, consequently, approximations of the semantics of logic programs. We have shown that the meaning function $T_{\mathcal{P}}$ of a logic program $\mathcal{P}$ can be approximated arbitrarily well by a 3–layered feed forward network if $\mathcal{P}$ is an acyclic logic programs that admits an injective level mapping. The approximation is computed in two time steps: propagation from the input to the hidden layer and propagation from the hidden layer to the output layer. Moreover, by turning the feed forward network into a recurrent one it was shown that it is also possible to approximate the least fixed point of $T_{\mathcal{P}}$, i.e. the semantics of $\mathcal{P}$, arbitrarily well.

To show that there exists a feed forward network that is capable of approximating a real valued function corresponding to $T_{\mathcal{P}}$ for a given program we have made use of a theorem given in [15], that states the existence of such a network if the real valued function is continuous. However, the theorem given in [15] (Theorem 3 in this article) does not state how the feed forward network looks like or how it can be constructed. Neither we got any clue how to construct a neural network for encoding or decoding a first order interpretation of a logic program into activation levels of the input unit.

In other words, whereas our mapping R enables us to prove the desired theoretical results, it does not seem to be a practical solution for representing the interpretations in the process of the iteration of $T_{\mathcal{P}}$. To solve this problem we are searching for suitable representations for the interpretations such that the approximation results are still valid. Obviously the naive solution to construct a network with input and output units each representing an atom, i.e. an element of the Herbrand base of $\mathcal{P}$, is not suitable because we do not know in advance which element we will really need to represent. We have to represent all the atoms of the Herbrand base, which are infinitely many in general.

A more desirable approach is to use fixed–length distributed descriptions to represent the elements of our domain. Models such as the (labeled) recursive auto associative memory [36, 44] or holographic reduced representation [35] provide such representations for term structures and, therefore, for atoms. As discussed in [22] however, these models lack the ability to handle structures of variable depth and, consequently, are not suitable for our problem. The problem of how to distributedly represent term structures in such a network is still unsolved.

If we restrict ourselves to a limited depth of the terms that can be handled during the computation, then these models can be applied to represent terms and atoms. But there is also the additional problem of how to represent interpretations, i.e. sets of atoms. These sets may consist of infinitely many elements, and this may happen after a single application of $T_{\mathcal{P}}$ to a finite interpretation. This would already exceed our finite memory after one computation step of $T_{\mathcal{P}}$. A solution to this problem is to use finite representations for the infinite interpretations occuring in such a computation as shown in [6]. However, the problem of how this approach can be realized in a connectionist system is not yet solved.

Taken all these arguments and open problems into account our future work will focus on the representational problem. Because the theoretical results formulated in this paper state the existence of a network to approximate the meaning function

$T_\mathcal{P}$, we strongly believe in the existence of such a representation.

But there is also another line of future research. The logic programs programs considered in this article were acyclic and had to admit an injective level mapping. These conditions are sufficient, but it is an open question whether they are necessary as well. The condition of being acyclic was used to show that $T_\mathcal{P}$ is a contraction on the metric space $(2^{B_\mathcal{P}}, d_\mathcal{P})$. As shown by Seda and Hitzler, acyclic logic programs are not the largest class that admit this property [38, 39]. The condition of admitting an injective level mapping was used to show that the mapping R is injective. Both conditions were used to show our main result, but we expect that they can be weakened.

Acknowledgements

We would like to thank the anonymous referees for their very valuable comments. Their suggestions helped to improve the article considerably.

Appendix

Proof of Proposition 1: We will show that $d_\mathcal{P}(T_\mathcal{P}(I), T_\mathcal{P}(J)) \leq \frac{1}{2}\, d_\mathcal{P}(I, J)$. Assume $d_\mathcal{P}(I, J) = 2^{-n}$, so that I and J agree on all ground atoms A with $|A| < n$. To show that $d_\mathcal{P}(T_\mathcal{P}(I), T_\mathcal{P}(J)) \leq 2^{-(n+1)}$ it is sufficient to show that $T_\mathcal{P}(I)$ and $T_\mathcal{P}(J)$ agree on all ground atoms A with $|A| < n+1$.

Now suppose that $T_\mathcal{P}(I)$ and $T_\mathcal{P}(J)$ disagree on a ground atom A with $|A| < n+1$. There are two cases that arise: (a) $A \in T_\mathcal{P}(I)$ and $A \notin T_\mathcal{P}(J)$ and (b) the other way around. Because case (b) is symmetric to (a) we omit its proof and concentrate on proving (a):

Because $A \in T_\mathcal{P}(I)$ and $A \notin T_\mathcal{P}(J)$ there is a ground instance $A \leftarrow L_1, \ldots, L_n$ of a clause in $\mathcal{P}$ such that $\{L_1, \ldots, L_n\} \subseteq I$ and $\{L_1, \ldots, L_n\} \not\subseteq J$. Consequently, there is a literal L_k with $k \in [1, n]$ such that $L_k \in I$ and $L_k \notin J$. But from the definition of acyclic logic programs we know that $|A| > |L_k|$. This is a contradiction to our assumption that I and J agree on all atoms with a level less than n, so the proposition is established. □

To improve the intelligibility of the article, Proposition 2 was presented before Proposition 3. The proofs are presented in inverted order because the proof of Proposition 2 depends on Proposition 3.

Proof of Proposition 3: Let n be the lowest level of an atom for which the interpretations I and J disagree. Thus $d_\mathcal{P}(I, J) = 2^{-n}$ and hence $d_\mathcal{P}(I, J)^2 = 4^{-n}$.

Since all atoms A with level $\|A\| < n$ are mapped to the same truth value by I and J it follows from the definition of R that

$$4^{-n} - \sum_{i=n+1}^{\infty} 4^{-i} \leq |R(I) - R(J)| \leq 4^{-n} + \sum_{i=n+1}^{\infty} 4^{-i} \ ,$$

which yields $\frac{2}{3} * 4^{-n} \leq |R(I) - R(J)| \leq \frac{4}{3} * 4^{-n}$ and hence the proposition holds. □

Proof of Proposition 2: To prove that the set $\mathcal{D}_f$ is closed we prove that the limits of sequences of elements of $\mathcal{D}_f$ are contained in $\mathcal{D}_f$.

Let $\mathcal{R} = r_1, r_2, \ldots, r_i \in \mathcal{D}_f$ be a convergent sequence on $\mathbb{R}$ and $r = \lim_{i \to \infty} r_i$ the limit $r \in \mathbb{R}$ of this sequence. Since $\mathbb{R}$ with Euclidean distance is a complete metric space, $\mathcal{R}$ is Cauchy. Consider now the sequence of interpretations $\mathcal{I} = I_1, I_2, \ldots$ such that $I_i = R^{-1}(r_i)$. This sequence is Cauchy as well: since $\mathcal{R}$ is Cauchy, for every $\varepsilon > 0$ there is an $N \in \mathbb{N}$ such that for every $n, m \geq N$ we have $|R(I_n) - R(I_m)| = |r_n - r_m| < \frac{2}{3}\varepsilon^2$, and thus, according to Proposition 3, $d_\mathcal{P}(I_n, I_m) < \varepsilon$. Consequently, $\mathcal{I}$ is convergent because $(2^{B_\mathcal{P}}, d_\mathcal{P})$ is a complete metric space.

Let I be the limit of $\mathcal{I}$. $R(I)$ is now the limit of $\mathcal{R}$: for every $\varepsilon > 0$ there is an $N \in \mathbb{N}$ such that for all $n \geq N$ we have that $d_{\mathcal{P}}(I_n, I) < \sqrt{\frac{3}{4}\varepsilon}$ and thus, according to Proposition 3, $|r_n - r| = |R(I_n) - R(I)| < \varepsilon$. Since $r = R(I)$ we have that $r \in \mathcal{D}_f$, and thus the proposition follows. □

Proof of Proposition 4: We distinguish four cases with respect to r and r':

(i) $r, r' \in \mathcal{D}_f$:
Let $I = R^{-1}(r)$ and $I' = R^{-1}(r')$. From Proposition 3 and since $I, I' \in 2^{B_{\mathcal{P}}}$ it follows

$$\frac{2}{3} d_{\mathcal{P}}(I, I')^2 \leq |R(I) - R(I')| = |r - r'| \tag{A1}$$

and

$$|f_{\mathcal{P}}(r) - f_{\mathcal{P}}(r')| = |R(T_{\mathcal{P}}(I)) - R(T_{\mathcal{P}}(I'))| \leq \frac{4}{3} d_{\mathcal{P}}(T_{\mathcal{P}}(I), T_{\mathcal{P}}(I'))^2 \ . \tag{A2}$$

Since according to the proof of Proposition 1 $T_{\mathcal{P}}$ is a contraction, i.e. $d_{\mathcal{P}}(T_{\mathcal{P}}(I), T_{\mathcal{P}}(I')) < d_{\mathcal{P}}(I, I')$, and by the definition of $d_{\mathcal{P}}$, $d_{\mathcal{P}}(T_{\mathcal{P}}(I), T_{\mathcal{P}}(I')) \leq \frac{1}{2} d_{\mathcal{P}}(I, I')$ we conclude $d_{\mathcal{P}}(T_{\mathcal{P}}(I), T_{\mathcal{P}}(I'))^2 \leq \frac{1}{4} d_{\mathcal{P}}(I, I')^2$. The proposition follows immediately using (A1) and (A2).

(ii) $r \notin \mathcal{D}_f$, $r' \in \mathcal{D}_f$:
According to Definition 3 we have to distinguish three cases:

(iia) $r > \max(\mathcal{D}_f)$: We can apply case (i) to show

$$|f_{\mathcal{P}}(\max(\mathcal{D}_f)) - f_{\mathcal{P}}(r')| \leq \frac{1}{2} |\max(\mathcal{D}_f) - r'| \ . \tag{A3}$$

Because $r' \leq \max(\mathcal{D}_f) < r$ we have $f_{\mathcal{P}}(r) = f_{\mathcal{P}}(\max(\mathcal{D}_f))$ and $|\max(\mathcal{D}_f) - r'| < |r - r'|$ and thus the proposition follows.

(iib) $r < \min(\mathcal{D}_f)$: The proof of this case is similar to the previous case.

(iic) $\min(\mathcal{D}_f) < r < \max(\mathcal{D}_f)$:
In the following we use the abbreviations $S_M = \frac{M_r - r}{M_r - m_r}$ and $S_m = \frac{r - m_r}{M_r - m_r}$. According to Definition 3 we have

$$f_{\mathcal{P}}(r) = S_M \ \bar{f}_{\mathcal{P}}(m_r) + S_m \ \bar{f}_{\mathcal{P}}(M_r) \tag{A4}$$

where $m_r = \max(\mathcal{D}_f \cap [0, r])$ and $M_r = \min(\mathcal{D}_f \cap [r, 1])$. Because of $m_r, M_r, r' \in \mathcal{D}_f$ we can apply case (i) and find that

$$|f_{\mathcal{P}}(m_r) - f_{\mathcal{P}}(r')| \leq \frac{1}{2} |m_r - r'| \qquad \text{and} \qquad |f_{\mathcal{P}}(M_r) - f_{\mathcal{P}}(r')| \leq \frac{1}{2} |M_r - r'| \ . \tag{A5}$$

Thus, we have

$$S_M \ |f_{\mathcal{P}}(m_r) - f_{\mathcal{P}}(r')| + S_m \ |f_{\mathcal{P}}(M_r) - f_{\mathcal{P}}(r')| \leq \frac{1}{2} \left(S_m \ |m_r - r'| + S_M \ |M_r - r'|\right) \ . \tag{A6}$$

Since $S_M + S_m = 1$ as well as the triangle inequality hold, the left side of (A6) is greater or equal to

$$| \ S_M \bar{f}_{\mathcal{P}}(m_r) + S_m \bar{f}_{\mathcal{P}}(M_r) - f_{\mathcal{P}}(r') \ | = |f_{\mathcal{P}}(r) - f_{\mathcal{P}}(r')|$$

Furthermore, r' cannot lie in the interval (m_r, M_r) because $r' \in \mathcal{D}_f$. Thus, the right side of (A6) is equal to

$$\frac{1}{2} \ | \ S_M m_r + S_m M_r - r' \ | = \frac{1}{2} |r - r'| \ ,$$

and thus the proposition follows.

(iii) $r \in \mathcal{D}_f$, $r' \notin \mathcal{D}_f$:
In analogy to case (ii).

(iv) $r \notin \mathcal{D}_f$, $r' \notin \mathcal{D}_f$:
The proof of this case is almost identical to the case (ii) except that we apply case(iii) instead of case(i) to establish (A3) and (A5). We have to split a fourth sub-case off the sub-case corresponding to (iic), though: If r' is in the interval (m_r, M_r) then we find that $m_r = m_{r'}$ and $M_r = M_{r'}$ and therefore

$$f_{\mathcal{P}}(r) = \frac{r - m_r}{M_r - m_r} f_{\mathcal{P}}(M_r) + \frac{M_r - r}{M_r - m_r} f_{\mathcal{P}}(m_r) \tag{A7}$$

and

$$f_{\mathcal{P}}(r') = \frac{r' - m_r}{M_r - m_r} f_{\mathcal{P}}(M_r) + \frac{M_r - r'}{M_r - m_r} f_{\mathcal{P}}(m_r) \ . \tag{A8}$$

Consequently, we have

$$|f_{\mathcal{P}}(r') - f_{\mathcal{P}}(r)| \ = | \ \frac{r' - r}{M_r - m_r} f_{\mathcal{P}}(M_r) - \frac{r' - r}{M_r - m_r} f_{\mathcal{P}}(m_r) \ | = | \ (r' - r) \frac{f_{\mathcal{P}}(M_r) - f_{\mathcal{P}}(m_r)}{M_r - m_r} \ | \ . \tag{A9}$$

Since we can apply case (i) to establish $|f_{\mathcal{P}}(M_r) - f_{\mathcal{P}}(m_r)| \leq \frac{1}{2}|M_r - m_r|$, the proposition follows from (A9). □

Proof of Theorem 4:
Let $f_{\mathcal{P}}$ be the continuous real valued function corresponding to $T_{\mathcal{P}}$. According to Theorem 3 there exists an feed forward network computing the function $\tilde{f}_{\mathcal{P}}$ which satisfies

$$\max_{r \in [-1,1]} |f_{\mathcal{P}}(r) - \tilde{f}_{\mathcal{P}}(r)| < \varepsilon \tag{A10}$$

with $\varepsilon = \frac{1}{3*2^{2N+2}}$. If we connect the input and output unit using a connection with weight 1, and start with an initial activation $x \in [-1, 1]$, we obtain a recursive network which computes $\tilde{f}^n_{\mathcal{P}}(x)$ in the n–th step.

Let M be the unique fixed point of $T_{\mathcal{P}}$. We prove now by induction that for every $n \in \mathbb{N}$

$$|\tilde{f}^n_{\mathcal{P}}(x) - R(M)| < \frac{1}{2^n} + 2\varepsilon \ . \tag{A11}$$

Since both the value of $R(M)$ and $\tilde{f}^0_{\mathcal{P}}(x) = x$ are within the interval $[-1, 1]$ we have that $|\tilde{f}^0_{\mathcal{P}}(x) - R(M)| < 1 + 2\varepsilon$. Thus, (A11) is fulfilled for $n = 0$.

Let the induction hypothesis be true for $n-1$, that is, $|\tilde{f}^{n-1}_{\mathcal{P}}(x) - R(M)| < \frac{1}{2^{n-1}} + 2\varepsilon$. From Proposition 3 follows

$$|f_{\mathcal{P}}(\tilde{f}^{n-1}_{\mathcal{P}}(x)) - f_{\mathcal{P}}(R(M))| < \frac{1}{2}\left(\frac{1}{2^{n-1}} + 2\varepsilon\right) = \frac{1}{2^n} + \varepsilon \ .$$

Considering $f_{\mathcal{P}}(R(M)) = R(M)$ and (A10) we get $|\tilde{f}_{\mathcal{P}}(\tilde{f}^{n-1}_{\mathcal{P}}(x)) - R(M)| < \frac{1}{2^n} + 2\varepsilon$ using the triangle inequality. By the induction principle, we thus have (A11) for all $n \in \mathbb{N}$.

Let $n_0 = 2N + 2$, $n \geq n_0$ and recall that $\varepsilon = \frac{1}{3*2^{2N+2}}$. By (A11) we get

$$|\tilde{f}^n_{\mathcal{P}}(x) - R(M)| \ < \ \frac{1}{2^n} + \frac{1}{3 * 2^{2N+2}} \ \leq \ \frac{1}{2^{2N+2}} + \frac{1}{3 * 2^{2N+2}} \ = \ \frac{1}{3 * 4^N} \ .$$

Because of the definition of R^{-1} the inequality $|R(R^{-1}(\tilde{f}^n_{\mathcal{P}}(x))) - \tilde{f}^n_{\mathcal{P}}(x)| < |R(M) - \tilde{f}^n_{\mathcal{P}}(x)|$ holds. (The nearest point to $\tilde{f}^n_{\mathcal{P}}(x)$ in $\mathcal{D}_f$ cannot be further away than $R(M) \in \mathcal{D}_f$.) By the triangle inequality we get

$$|R(R^{-1}(\tilde{f}^n_{\mathcal{P}}(x))) - R(M)| \ \leq \ 2 \ |\tilde{f}^n_{\mathcal{P}}(x) - R(M)| \ < \ \frac{2}{3 * 4^N}$$

and by the application of Proposition 3 the theorem follows. □

Notes

1. A parallel computational model requiring $p(n)$ processors and $t(n)$ time to a solve a problem of size n is *optimal* if $p(n) \times t(n) = O(T(n))$, where $T(n)$ is the sequential time to solve this problem.
2. Under the name of ω–locally hierarchical programs.
3. See [37] for a definition of perfect models.
4. See [10] for a definition of the completion of a logic program.
5. In this article we do not use the more general result of [20] that a feed forward network can approximate Borel–measurable arbitrarily well because the notion of approximation in [20] includes only *almost all* points of the domain of the function instead of *all* points, as in the case of [15].
6. The obvious choice of the binary number system does not work because of the ambiguity $0.1000\ldots_2 = 0.01111\ldots_2$. Additionally, the quaternary number system ensures $f_{\mathcal{P}}$ being a contraction (see Proposition 4).
7. There is at most one such atom because $||\ ||$ is injective.
8. The quaternary representation of $\frac{1}{3}$ is $0.1111111\ldots_4$.

References

1. N. Alon and A. Dewdney and T. Ott. Efficient simulation of finite automata by neural nets. *Journal of the Association for Computing Machinery*, 38(2), pp. 495–514, 1991.
2. R. Andrews, J. Diederich, and A. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge–Based Systems*, 8(6), 1995.
3. K.R. Apt and M. Bezem. Acyclic Programs. In D.H.D. Warren and P. Szeredi, editors, *Logic Programming*, Proceedings of the Seventh International Conference, Jerusalem, Israel, June 18-20, MIT Press pp. 617–633, 1990.
4. K.R. Apt and M. Bezem. Acyclic Programs. *New Generation Computing*, 9(3/4), pp. 335–363, 1991.
5. K.R. Apt and M.H. Van Emden. Contributions to the Theory of Logic Programming. *Journal of the ACM*, 29, pp. 841–862, 1982.
6. S.–E. Bornscheuer. Generating Rational Models. In M. Maher, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming (JICSLP)*, p. 547. MIT Press, 1996.
7. Sven-Erik Bornscheuer, Steffen Hölldobler, Yvonne Kalinke, and Antje Strohmaier. *Massively Parallel Reasoning*, volume II of *Automated Deduction — A Basis for Applications*, chapter 11, pages 291–321. Kluwer Academic Publishers, 1998.
8. M. Casey. The dynamics of discrete–time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, 8(6), pp. 1135–1178, 1996.
9. L. Cavedon. Acyclic programs and the completeness of SLDNF–resolution. *Theoretical Computer Science*, 86(1), pp. 81–92, 1991.
10. K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, Plenum, New York, NY, pp. 293–322, 1978.
11. A.S. d'Avila Garcez and G. Zaverucha and L.A.V. de Carvalho. Logic programming and inductive learning in artificial neural networks. In Ch. Herrmann and F. Reine and A. Strohmaier, editors, *Knowledge Representation in Neural Networks*, pp. 33–46, Berlin, Logos Verlag, 1997.
12. M. Fitting. Metric methods – three examples and a theorem. *Journal of Logic Programming*, 21(3), pp. 113–127, 1994.
13. P. Frasconi and M. Gori and M. Maggini and G. Soda. Representation of finite state automata in recurrent radial basis function networks. *Machine Learning*, 23, pp. 5–32, 1996
14. M. Fujita, R. Hasegawa, M. Koshimura and H. Fujita. Model Generation Theorem Provers on a Parallel Inference Machine. In ICOT Staff, editors, *Fifth Generation Computer Systems '92: Proceedings of the International Conference on Fifth Generation Computer Systems*, IOS Press, pp. 357–375, 1992.
15. K.–I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2, pp. 183–192, 1989.
16. C.L. Giles and C. Miller and G.Z. Sun and H.H. Chen and Y.C. Lee and D. Chen. Learning and extracting finite state automata with second–order recurrent neural networks. *Neural Computation*, 4(3), pp. 393–405, 1992.
17. S. Hölldobler. Automated inferencing and connectionist models. Technical Report AIDA–93–06, Intellektik, Informatik, TH Darmstadt, 1993. (Postdoctoral Thesis).
18. S. Hölldobler and Y. Kalinke. Towards a massively parallel computational model for logic programming. In *Proceedings of the ECAI94 Workshop on Combining Symbolic and Connectionist Processing*, pp. 68–77, ECCAI, 1994.
19. S. Hölldobler and Y. Kalinke and H.–P. Störr. Recurrent Neural Networks to Approximate the Semantics of Acceptable Logic Programs. In G. Antoniou and J. Slaney, editors, *Advanced Topics in Artificial Intelligence*, LNAI 1502, Springer–Verlag, 1998.
20. K. Hornik, M. Stinchcombe and H. White. Multilayer feedforward networks are universal approximators. *Neuronal Networks*, 2, pp. 359–366, 1989.
21. P.N. Johnson–Laird and R.M.J. Byrne. *Deduction*. Lawrence Erlbaum Associates, Hove and London (UK), 1991.
22. Y. Kalinke. Using Connectionist Term Representation for First–Order Deduction – A Critical View. In F. Maire, R. Hayward and J. Diederich, editors, *Connectionist Systems for Knowledge Representation Deduction*, Queensland University of Technology, 1997.
23. Y. Kalinke and H. Lehmann. Computation in Recurrent Neural Networks: From Counters to Iterated

Function Systems. In G. Antoniou and J. Slaney, editors, *Advanced Topics in Artificial Intelligence*, LNAI 1502, Springer–Verlag, 1998.

24. J.F. Kolen. *Exploring the Computational Capabilities of Recurrent Neural Networks.* PhD Thesis, Ohio State University, 1994.
25. S.C. Kremer. Finite state automata that recurrent cascade–corelation cannot represent. In D.S. Touretzky, et al., editors, *Advances in Neural Information Processing Systems 8*, MIT Press, Cambridge, MA, pp. 612–618, 1996.
26. T. E. Lange and M. G. Dyer. High-level inferencing in a connectionist network. *Connection Science*, 1:181 – 217, 1989.
27. J. W. Lloyd. *Foundations of Logic Programming.* Springer, 1987.
28. R. Manthey and F. Bry. SATCHMO: A Theorem Prover Implemented in Prolog. In E. Lusk and R. Overbeek, editors, *Proceedings of the Conference on Automated Deduction*, LLNCS 310, pp. 415–434. Springer, 1988.
29. J. McCarthy. Epistemological challanges for connectionism. *Behavioural and Brain Sciences*, 11:44, 1988. Commentary to [43].
30. C.W. Omlin and C.L. Giles. Constructing deterministic finite–state automata in recurrent neural networks. *Journal of the ACM*, 45(6), pp. 937–972, 1996.
31. C.W. Omlin and C.L. Giles. Extraction of rules from discrete-time recurrent neural networks. *Neural Networks*, 9(1), pp. 41–52, 1996.
32. G. Pinkas. Expressing first-order logic in symmetric connectionist networks. In L. N. Kanal and C. B. Suttner, editors, *Informal Proceedings of the International Workshop on Parallel Processing for AI*, pages 155–160, Sydney, Australia, August 1991 1991.
33. G. Pinkas. Symmetric neural networks and logic satisfiability. *Neural Computation*, 3, 1991.
34. T. A. Plate. Holographic reduced representations. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 30–35, 1991.
35. T. A. Plate. *Distributed Representations and Nested Compositional Structure.* PhD thesis, Department of Computer Science, University of Toronto, 1994.
36. J. B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46:77–105, 1990.
37. T.C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In J. Minker, editor, *Foundations of deductive databases and logic programming*, Morgan Kaufmann Publishers Inc., Los Altos, pp. 193–216, 1988.
38. A.K. Seda and P. Hitzler. Topology and Iterates in Computational Logic. In *Topology Proceedings Vol.II*, Proceedings of the Twelveth Summer Conference on General Topology and its Applications: Special Session on Topology in Computer Science, Annals of the New York Academy of Sciences, Ontario, August 1997 (to appear).
39. A.K. Seda and P. Hitzler. Strictly Level-Decreasing Logic Programs. In A. Butterfield and S. Flynn, editors, *Proceedings of the Second Irish Workshop on Formal Methods*, Electronic Workshops in Computing, Springer–Verlag 1998 (to appear).
40. L. Shastri and V. Ajjanagadde. From associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioural and Brain Sciences*, 16(3):417–494, September 1993.
41. H. Siegelmann and E.D. Sontag. Turing Computability with Neural Nets. *Applied Mathematics Letters*, 4(6), pp. 77–80, 1991.
42. J. Slaney. Scott: A model–guided theorem prover. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 109–114, 1993.
43. P. Smolensky. On the Proper Treatment of Connectionism. *Behavioral and Brain Sciences*, 11(1), pp. 1–23, 1988.
44. A. Sperduti. Labeling RAAM. Technical Report TR-93-029, International Computer Science Institute, Berkeley, CA, 1993.
45. A. Sperduti. On the computational power of recurrent neural networks. *Neural Networks*, 10(3), pp. 395–400, 1997.
46. G.G. Towell and J.W. Shavlik. Extracting Refined Rules from Knowledge-Based Neural Networks. *Machine Learning*, 13(1), pp. 71–101, 1993.
47. R.L. Watrous and G. Kuhn. Induction of Finite-State Automata Using Second–Order Recurrent Networks. In J.E. Moody, et al., editors, *Neural Information Processing Systems 4*, Morgan Kaufmann, San Mateo, CA, pp. 309–316, 1992.
48. S. Willard. *General Topology.* Addison–Wesley, 1970.

Steffen Hölldobler Steffen Hölldobler has received his doctorate in science at the University of the Armed Forces, Munich, in 1988. After a postdoctoral year at the International Computer Science Institute, Berkeley, and an associate professorship at the Darmstadt University of Technology he is a professor for knowledge representation and reasoning at the Dresden University of Technology since 1993.

Yvonne Kalinke Yvonne Kalinke received a MSc. degree in computer science from Dresden University of Technology (Germany) in 1994 and is now a Ph.D. student in computer science at the same university. She is currently staying at the Machine Learning Research Centre of Queensland University of Technology, Brisbane (Australia). Her thesis is on connectionist term representations and a connectionist system for model generation for logic programs. Other research interests include the relation between recurrent neural networks and dynamical systems and the semantics of logic programs.

Hans-Peter Störr Text of biography...