Recurrent Neural Networks to Approximate the Semantics of Acceptable Logic Programs

Steffen Hölldobler [1] and Yvonne Kalinke [2] * and Hans–Peter Störr [1]

[1] Computer Science Department
University of Technology Dresden
D-01062 Dresden, Germany
{sh,haps}@inf.tu-dresden.de

[2] Neurocomputing Research Centre
Queensland University of Technology
Brisbane, Australia, GPO Box 2434, QLD 4001
yvonne@fit.qut.edu.au

Abstract. In [9] we have shown how to construct a 3–layer recurrent neural network (RNN) that computes the iteration of the meaning function $T_{\mathcal{P}}$ of a given propositional logic program, what corresponds to the computation of the semantics of the program.
In this paper we define a notion of approximation for interpretations and prove that there exists a feed forward neural network (FNN) that approximates the calculation of $T_{\mathcal{P}}$ for a given (first order) acceptable logic program with an injective level mapping arbitrarily well. By extending the FNN by recurrent connections we get a RNN whose iteration approximates the fixed point of $T_{\mathcal{P}}$.
The proof is found by taking advantage of the fact that for acceptable logic programs, $T_{\mathcal{P}}$ is a contraction mapping on the complete metric space of the interpretations for the program. Mapping this metric space to the metric space $\mathbb{R}$ the real valued function $f_{\mathcal{P}}$ corresponding to $T_{\mathcal{P}}$ turns out to be continuous and a contraction and for this reason can be approximated by an indicated class of FNN.

1 Introduction

Many researchers are convinced that intelligent agents reason by generating models and deducing conclusions with respects to these models (see e.g. [11]). 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 one of the major open question is how the full power of deductive processes can be implemented on connectionist networks (see e.g. [16]). In this paper we will focus on this problem by establishing a strong link between model generation in the

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

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. [13, 7, 15]). 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. [12]), 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 [1], or the class of acceptable programs, where the just mentioned correspondence can be shown using metric methods [6].

It turned out that the computation of the least model for a program from one of the just mentioned classes can be performed by a recursive network of binary threshold units if the programs are propositional [9]: 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 respective output layer of a 3–layered feed forward network, or FNN for short. 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. Figure 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 (RNN) by connecting corresponding units in the output and input layer with weight 1 allows to compute the least fixed point of $T_{\mathcal{P}}$ for acceptable logic programs.

One should observe that the number of units and the number of connections in an RNN 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.[3]

The approach in [9] provides a new computational model for the computation of the least model of logic programs, which is massively parallel and optimal. Since FNNs are widely used and there are powerful learning algorithms like backpropagation, we also may use these techniques to adapt our program. We

[3] 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.

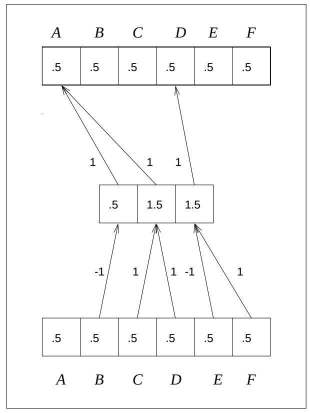

Fig. 1. A FNN computing $T_{\mathcal{P}}$ for the program $\mathcal{P} = \{A \leftarrow \neg B;\ A \leftarrow C, D;\ D \leftarrow \neg E, F\}$. The numbers in the units denote thresholds whereas the numbers at the connections denote weights. Connections with weight 0 are not shown.

can, for instance, train a given FNN with knowledge not yet included in the program. If we then extract a new logic program from the trained network we get an extended program including new clauses (see [5]). Moreover, the approach establishes a strong relationship between propositional logic programming and recursive neural networks. Since, in contrast to RNNs, logic programs are well understood this may help us to gain a better insight into RNNs, to formally analyze these networks and to give a declarative semantics for what these networks are doing.

But the question remains how this relationship may look like in the first order case? Since in this case an interpretation may be an infinite subset of the set of ground atoms wrt the alphabet underlying a program, the construction developed in [9] may lead to infinite FNNs and RNNs if 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.

FNNs 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 [8, 10]. 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 an FNN approximating $f_{\mathcal{P}}$ to a desired degree of accuracy, can we then turn the FNN into an RNN approximating the least fixed point of $T_{\mathcal{P}}$ arbitrarily well?

In this paper 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 paper is organized as follows. After stating some preliminaries concerning logic programs and metric spaces in Section 2, we show in Section 3 how to encode the domain of the meaning function $T_{\mathcal{P}}$ into $\mathbb{R}$. In Section 4 we use this mapping to construct a real valued function $f_{\mathcal{P}}$ corresponding to the function $T_{\mathcal{P}}$. In Section 5 we show that the real valued function $f_{\mathcal{P}}$ is continuous for a certain class of programs and that an FNN with sigmoidal activation functions and at least one hidden layer can approximate the function $f_{\mathcal{P}}$. Thereafter we show how to extend the FNN to a RNN such that the iteration of $f_{\mathcal{P}}$ corresponds to the iteration of $T_{\mathcal{P}}$ and therefore computes the least fixed point of $T_{\mathcal{P}}$. Since in the case of acceptable programs such a fixed point does always exist we end up with a RNN computing the semantics of the logic program $\mathcal{P}$. Finally, we discuss our results and point out future work in Section 8.

2 Preliminaries

In the following two subsections we briefly recall basic notions and notations concerning logic programs according to [12] and metric spaces as in [6].

2.1 Logic Programs

A *logic program* $\mathcal{P}$ is a collection of clauses of the form $A \leftarrow L_1, \ldots, L_n$, where $n \geq 0$, A is an atom and L_i, $1 \leq i \leq n$, are literals. A program $\mathcal{P}$ is called *definite* if each clause occurring in $\mathcal{P}$ contains only atoms. $B_{\mathcal{P}}$ denotes the *Herbrand base* wrt the alphabet underlying $\mathcal{P}$. A *level mapping* for a program $\mathcal{P}$ is a function $| \, | : B_{\mathcal{P}} \rightarrow \mathbb{N}$, 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.

An *interpretation* is a mapping from ground atoms to $\{\mathbf{1}, \mathbf{0}\}$[4]. It is extended to literals, clauses and programs in the usual 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 $A \leftarrow L_1, \ldots, L_n$ in $\mathcal{P}$ such that $\bigwedge_{i=1}^{n} L_i \in I$.

Let $\mathcal{P}$ be a program and p and q predicate symbols. p *refers to* q if there is a clause in $\mathcal{P}$ with head p and q in its body. p *depends on* q if either $p = q$ or there is a sequence $p = p_1, \; p_2, \; \ldots, \; p_n = q$, where each p_i refers to p_{i+1}, $1 \leq i < n$. $Neg_{\mathcal{P}}$ is the set of atoms in $\mathcal{P}$ which occur in a negative literal in the body of a clause of $\mathcal{P}$. $Neg^*_{\mathcal{P}}$ is the set of atoms in $\mathcal{P}$ on which the variables in $Neg_{\mathcal{P}}$ depend. $\mathcal{P}^-$ is the set of clauses in $\mathcal{P}$ whose head contains a relation from $Neg^*_{\mathcal{P}}$.

[4] We use the usual notation of I: An atom $A \in B_{\mathcal{P}}$ is mapped to *true* iff $A \in I$.

Let $Comp(\mathcal{P})$ denote the Clark completion of a program $\mathcal{P}$ [3]. Furthermore, let $\mathcal{P}$ be a program, $|\ |$ a level mapping for $\mathcal{P}$, and I a model of $\mathcal{P}$ whose restriction to the atoms in $Neg^*_{\mathcal{P}}$ is a model for $Comp(\mathcal{P}^-)$. $\mathcal{P}$ is *acceptable* wrt $|\ |$ and I if for every ground instance of a clause $A \leftarrow L_1, \dots, L_n$ in $\mathcal{P}$ and for every i, $1 \leq i \leq n$, we find that $I \models \bigwedge_{j=1}^{i-1} L_j$ implies $|A| > |L_i|$. $\mathcal{P}$ is *acceptable* if it is acceptable wrt some level mapping and some model.

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 $\mathcal{S} = s_1,\ s_2,\ \dots,\ s_i \in \mathcal{M}$, 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 $\forall \varepsilon > O : \exists N : \forall n,m \geq N : d(s_n, s_m) \leq \varepsilon$. $(\mathcal{M}, d)$ 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 $\mathcal{P}$. The 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 [6] the distance function $d_{\mathcal{P}}$ associated with a level mapping $|\ |$ for $\mathcal{P}$ is actually a metrics and the metric space $(2^{B_{\mathcal{P}}}, d_{\mathcal{P}})$ is complete.

Proposition 1 (Fitting [6]). *Let $\mathcal{P}$ be an acceptable program. There exists a level mapping $|\ |$ such that the meaning function $T_{\mathcal{P}}$ is a contraction on the complete metric space $(2^{B_{\mathcal{P}}}, d_{\mathcal{P}})$, where $d_{\mathcal{P}}$ denotes the distance function associated with the level mapping $|\ |$.*

Theorem 2 (Banach Contraction Theorem [18]). *A contraction mapping f on a complete metric space has a unique fixed point. The sequence x, $f(x)$, $f(f(x)), \dots$ converges to this fixed point for any x.*

3 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 [9], where each ground atom is represented by a binary threshold unit 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 an FNN, 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 [8] that continuous functions can be approximated arbitrarily well by an FNN.[5] As we shall see, by restricting ourselves to a certain class of acceptable 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 as for instance used in [6]. We further require it to be injective, and express this by renaming it to $\|\ \|$. This restriction is discussed further in Section 6.

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 the real numbers:

Definition 3. The mapping $R : 2^{B_{\mathcal{P}}} \to \mathbb{R}$ is defined as $R(I) = \sum_{A \in I} 4^{-\|A\|}$.

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. 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 $I \in 2^{B_{\mathcal{P}}}$ [8].

There is a close relation between the metric $d_{\mathcal{P}}$ on interpretations and the distance of the corresponding real numbers:

Proposition 4. *For two interpretations $I, J \in 2^{B_{\mathcal{P}}}$ the following holds:*

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

Proof. Let n be the lowest level of an atom for which the interpretations I and J differ. 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. □

[5] In this paper we do not use the more general result of [10] that a FNN can approximate Borel–measurable arbitrarily well because the notion of approximation in [10] includes only *almost all* points of the domain of the function instead of *all* points, as in the case of [8].

[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 Prop. 6).

[7] There is at most one such atom because $\|\ \|$ is injective.

[8] Such a point exists, because the set $\{R(I) | I \in 2^{B_{\mathcal{P}}}\}$ is closed. If there are two such points, we take the lower one.

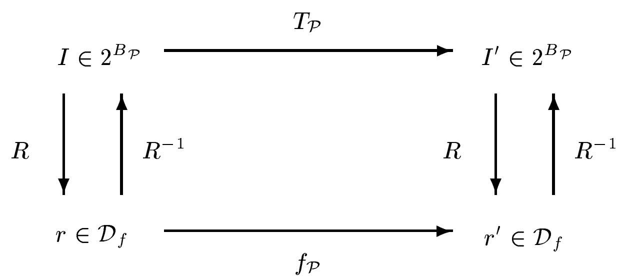

Fig. 2. The relation between $T_{\mathcal{P}}$ and $f_{\mathcal{P}}$.

4 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 $f_{\mathcal{P}}$. Please note that the encoding R defined in Definition 3 is just an example for mapping $2^{B_{\mathcal{P}}}$ to $\mathbb{R}$. We can use any mapping such that $f_{\mathcal{P}}$ is continuous on $\mathbb{R}$ iff $T_{\mathcal{P}}$ is a contraction on $2^{B_{\mathcal{P}}}$.

Definition 5. Let $\mathcal{P}$ be a logic program, $B_{\mathcal{P}}$ the 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 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:

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

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

To prove that $f_{\mathcal{P}}$ is continuous we first prove the following property of $f_{\mathcal{P}}$:

Proposition 6. *Let $\mathcal{P}$ be an acceptable 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 $\mathbb{R}$, since*

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

Proof. For any $r \in \mathbb{R}$ we can show that $I = R^{-1}(r) \in 2^{B_\mathcal{P}}$ and moreover define $f_\mathcal{P}$ by $f_\mathcal{P} : \mathbb{R} \to \mathbb{R} : r \mapsto R(T_\mathcal{P}(I))$ with $I = R^{-1}(x_r)$, where $x_r \in \mathcal{D}_f$ is determined by r [9]. Now let $r, r' \in \mathbb{R}$, $I = R^{-1}(r)$ and $I' = R^{-1}(r')$. From Proposition 4 and since $I, I' \in 2^{B_\mathcal{P}}$ it follows

$$2/3 * d_\mathcal{P}(I, I')^2 \leq |R(I) - R(I')| = |r - r'| \quad \text{and} \tag{1}$$

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

Since by 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 $d_\mathcal{P}(T_\mathcal{P}(I), T_\mathcal{P}(I'))^2 \leq \frac{1}{4} d(I, I')^2$. The proposition follows immediately using (1) and (2). □

As an immediate consequence of Proposition 6 we obtain:

Corollary 7. *Let $\mathcal{P}$ be an acceptable 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 the function $f_\mathcal{P}$ is continuous.*

5 Approximating the Meaning Function $T_\mathcal{P}$

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

Theorem 8 (Funahashi [8]). *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 real valued function on K. Then for an arbitrary $\varepsilon > 0$, there exists an integer N and real constants c_i, $\theta_i (i = 1, \ldots, N)$, $w_{ij} (i = 1, \ldots, N, j = 1, \ldots, n)$ 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 $\max_{\mathbf{x} \in K} |f(x_1, \ldots, x_n) - \tilde{f}(x_1, \ldots, x_n)| < \varepsilon$.

In other words, for an arbitrary $\varepsilon > 0$, there exists a 3–layered FNN whose output functions for the hidden layer are $\phi(x)$, 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 $\max_{\mathbf{x} \in K} |f(x_1, \ldots, x_n) - \tilde{f}(x_1, \ldots, x_n)| < \varepsilon$.

Applying Theorem 8 to Corollary 7 which states that $f_\mathcal{P}$ is continuous we can easily show that the following theorem holds:

Theorem 9. *Let $\mathcal{P}$ be an acceptable logic program with an injective level mapping and $T_\mathcal{P}$ the meaning function associated with $\mathcal{P}$. Let $f_\mathcal{P}$ be the continuous real valued function corresponding to $T_\mathcal{P}$. Then for an arbitrary $\varepsilon > 0$, there exists an FNN 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* $\max_{x \in [0,1]} |f_\mathcal{P}(x) - \tilde{f}_\mathcal{P}(x)| < \varepsilon$.

[9] We omit details due to space limitations.

The theorem states that given a certain accuracy we can construct an FNN that approximates $f_{\mathcal{P}}$ to this desired degree of accuracy. Using the mapping R^{-1} we thereby obtain an FNN capable of approximating the meaning function $T_{\mathcal{P}}$ for an acceptable logic program $\mathcal{P}$ with injective level mapping. In the following section we exemplify this result and clarify the notion of an approximation of $T_{\mathcal{P}}$.

6 An Example

Consider the program

$$\mathcal{P}: \quad p(0). \\ p(s(X)) \leftarrow p(X).$$

and the injective level mapping $\|p(s^n(0))\| = n+1$ for $n \in \mathbb{N}$.[10] The program is acceptable wrt the level mapping $\|\|$ and the interpretation $I = B_{\mathcal{P}}$. In this case we find

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

The iteration of $T_{\mathcal{P}}$, which yields the semantics of the program $\mathcal{P}$, corresponds to 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 $[\frac{a-\frac{1-4\varepsilon}{3}}{4} + \frac{1-4\varepsilon}{3}, \frac{b-\frac{1+4\varepsilon}{3}}{4} + \frac{1+4\varepsilon}{3}]$. It is easy to see that in the limit the iteration of $\tilde{f}_{\mathcal{P}}$ yields a value within $[\frac{1-4\varepsilon}{3}, \frac{1+4\varepsilon}{3}]$ (though 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 model $\{p(0), p(s(0)), p(s(s(0))), \ldots\}$ of $\mathcal{P}$. The values for $p(s^n(0))$ wrt $R^{-1}(r)$ with $n \geq \log_4 \varepsilon - 1$ may differ from the intended model.

This clarifies our notion of approximation of an interpretation or a model: An interpretation I *approximates* an interpretation J to a degree N , if for all atoms $A \in B_{\mathcal{P}}$ with $||A|| < N$, $A \in I$ iff $A \in J$. Equivalently, $d_{\mathcal{P}}(I, J) \leq 2^{-N}$.

One should observe that in our example we are not only able to approximate the calculation of $T_{\mathcal{P}}$ but are able to approximate the fixed point of $T_{\mathcal{P}}$ by iteration of the approximation.

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

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

It is acceptable wrt the level mapping $|| : \forall X \in U_{\mathcal{P}}.\ p(f(X)) \mapsto 1, q \mapsto 2$ and the interpretation $I = \{\}$, but it does not admit an injective level mapping for

[10] $s^n(0)$ is an abbreviation for $\underbrace{s(s(\cdots s(}_{n}0)\cdots))$.

which $T_{\mathcal{P}}$ is a contraction because the level of q has to be greater than the maximum of the level of all infinitely ground instances of $p(f(X))$.

On the other hand, the program $\mathcal{P}$ given above consists of a binary recursive definite clause and a fact only. As shown in [4] this class of programs has the same computational power as a Turing machine. Since it was also shown that each definite program can automatically be transformed into such an *universal* program, the computational power of the class of acceptable logic programs that are considered here is not restricted.

7 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 9 we have learned that there exits an FNN computing $\tilde{f}_{\mathcal{P}}$, which is the approximation of $f_{\mathcal{P}}$, which in turn represents $T_{\mathcal{P}}$. Given such an FNN we could turn this FNN into an RNN by adding recurrent connections with weight 1 between corresponding units in the output and input layer of the FNN. Does such an RNN approximate the least fixed point of $T_{\mathcal{P}}$? Whereas Theorem 9 tells us the activation value $\tilde{f}_{\mathcal{P}}(r)$ is within ε of $f_{\mathcal{P}}(r)$ after the first iteration of the RNN, this small difference could lead to bigger difference each step. Our final theorem tells us this is not the case with our encoding R:

Theorem 10. *Let $\mathcal{P}$ be an acceptable logic program with an injective level mapping, $T_{\mathcal{P}}$ the meaning function associated with $\mathcal{P}$ and M the unique fixed point of $T_{\mathcal{P}}$. For an arbitrary $N \in \mathbb{N}$ there exists an RNN 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 $n \in \mathbb{N}$ with*

$$d_{\mathcal{P}}(R^{-1}(\tilde{f}_{\mathcal{P}}^{n}(0)), M) \leq \frac{1}{2^N} \ .$$

Proof. Let $f_{\mathcal{P}}$ be the continuous real valued function corresponding to $T_{\mathcal{P}}$. According to Theorem 9 there exists an FNN computing the function $\tilde{f}_{\mathcal{P}}$ which satisfies

$$\max_{r\in[0,1]} |f_{\mathcal{P}}(r) - \tilde{f}_{\mathcal{P}}(r)| < \varepsilon \tag{3}$$

with $\varepsilon = \frac{1}{2^{2N+3}}$. If we connect the input and output unit using a connection with weight 1, and start with an initial activation 0, we obtain an RNN which computes $\tilde{f}_{\mathcal{P}}^{n}(0)$ in the n-th step.

Let M be the unique fixed point of $T_{\mathcal{P}}$. Since the value of $f_{\mathcal{P}}$ is within the interval $[0,1]$ and $\tilde{f}_{\mathcal{P}}$ differs at most by ε, $|\tilde{f}_{\mathcal{P}}^{0}(0) - R(M)| < 1 + 2\varepsilon$ holds. Assume now $|\tilde{f}_{\mathcal{P}}^{n-1}(0) - R(M)| < \frac{1}{2^{n-1}} + 2\varepsilon$. From (3) follows $|f_{\mathcal{P}}(\tilde{f}_{\mathcal{P}}^{n-1}(0)) - f_{\mathcal{P}}(R(M))| < \frac{1}{2^n} + \varepsilon$. Considering $f_{\mathcal{P}}(R(M)) = R(M)$ and (3) we get $|\tilde{f}_{\mathcal{P}}(\tilde{f}_{\mathcal{P}}^{n-1}(0)) - R(M)| < \frac{1}{2^n} + 2\varepsilon$ and thus by induction

$$|\tilde{f}_{\mathcal{P}}^{n}(0) - R(M)| < \frac{1}{2^n} + 2\varepsilon \ . \tag{4}$$

If we take $n = 2N+2$ and insert $\varepsilon = \frac{1}{2^{2N+3}}$ we get $|\tilde{f}^n_{\mathcal{P}}(0) - R(M)| < \frac{1}{3*2^{2N}}$. Because of the definition of R^{-1} $|R(R^{-1}(\tilde{f}^n_{\mathcal{P}}(0))) - \tilde{f}^n_{\mathcal{P}}(0)| < \frac{1}{3*2^{2N}}$ holds[11]. Thus, $|R(R^{-1}(\tilde{f}^n_{\mathcal{P}}(0))) - R(M)| < \frac{2}{3*2^{2N}}$ and by the application of Proposition 4 the theorem follows. □

This theorem tells us that $R^{-1}(\tilde{f}^n_{\mathcal{P}}(0))$ and M agree in atoms of a level less than N. In other words, we can approximate the fixed point of the meaning function $T_{\mathcal{P}}$ of an acceptable logic program with an injective level mapping arbitrarily well with a recurrent neural network.

8 Discussion and Future Work

Extending the result of [9] for the propositional case, we have proven in our paper that it is possible to approximate the meaning function $T_{\mathcal{P}}$ arbitrarily well in constant time[12] by an 3–layered FNN (Theorem 9) and that it it is possible to approximate the fixed point of $T_{\mathcal{P}}$ by forming the FNN into an RNN (Theorem 10).

However, Theorem 9 does not give any clue as to how to construct such a network and how to represent the arguments of the computations, i.e. sets of atoms. While our mapping R enables us to prove these theoretical results, it does not seem to be a practical solution for representing the interpretations in the process of the iteration of $T_{\mathcal{P}}$ because of its brittleness. It does not exploit the natural insensitivity of FNN to disturbances.

A very obvious and easy solution avoiding these problems is to construct a network with input and output nodes each representing an atom, i.e. an element of the Herbrand base of $\mathcal{P}$. But because we do not know in advance which element we will really need to represent the result of the computation, i.e. the special interpretations, we have to represent all the atoms of the Herbrand base up to the end of our memory and it may happen, that we will not need many of them for our computation.

A solution to this problem is to use a reduced description to represent the elements of our domain, such as LRAAM– or HRR– coded terms (Labeled Recursive Auto Associative Memory [17], Holographic Reduced Representation [14]). The problem with the LRAAM model will then be that we have to train it with many of the terms we will need during the computation. Since we do not know which terms we will need during the computation the same problem will arise as before. A better solution is to use HRR–coded terms. We do not need a training before the computation and we will be able to save the used terms at the time they appear in the computation process. Then we can save them in a special memory which will allow us to reduce the decoding error and to save only the terms we really need during the computation.

---

11 The nearest point to $\tilde{f}^n_{\mathcal{P}}(0)$ in $\{A(I) | I \in 2^{B_{\mathcal{P}}}\}$ cannot be further away than $R(M)$.

12 For an interpretation I, $T_{\mathcal{P}}(I)$ is computed in two time steps: propagation from the input to the hidden layer and propagation from the hidden layer to the output layer.

Another problem in the practical application of the iteration of $T_{\mathcal{P}}$ with RNN is that the interpretations that must be represented can become infinite during one step of the computation of the function $T_{\mathcal{P}}$. This will fill our finite memory after one step of computing $T_{\mathcal{P}}$. A solution to this problem is to use finite representations for infinite interpretations as shown in [2].

References

1. K.R. Apt and M.H. Van Emden. *Contributions to the Theory of Logic Programming.* Journal of the ACM, **29**, pp. 841–862, 1982.
2. 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.
3. K. L. Clark. Negation as failure. In Gallaire and Nicolas, editors, *Workshop Logic and Databases*, CERT, Toulouse, France, 1977.
4. P. Devienne and P. Lebégue and A. Parrain and J. C. Routier and J. Würz. Smallest Horn Clause Programs. *Journal of Logic Programming*, **19**(20): pp. 635–679, 1994.
5. A.S. d'Avila Garcez, G. Zaverucha, and L.A.V. de Carvalho. Logic programming and inductive learning in artificial neural networks. In Ch. Herrmann, F. Reine, and A. Strohmaier, editors, *Knowledge Representation in Neural Networks*, pp. 33–46, Berlin, Logos Verlag, 1997.
6. M. Fitting. Metric methods – three examples and a theorem. *Journal of Logic Programming*, **21**(3), pp. 113–127, 1994.
7. M. Fujita and R. Hasegawa and M. Koshimura and H. Fujita. *Model Generation Theorem Provers on a Parallel Inference Machine.* Proceedings of the International Conference on Generation Computer Systems, 1992.
8. K.–I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, **2**, pp. 183–192, 1989.
9. 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.
10. K. Hornik and M. Stinchcombe and H. White. *Multilayer feedforward networks are universal approximators.* Neuronal Networks, 2, pp. 359–366, 1989.
11. P.N. Johnson–Laird and R.M.J. Byrne. *Deduction.* LEA, Hove and London, 1991.
12. J. W. Lloyd. *Foundations of Logic Programming.* Springer, 1987.
13. R. Manthey and F. Bry. SATCHMO: A Theorem Prover Implemented in Prolog. In: E. Lusk and R. Overbeek, editors, LLNCS 310, Springer, pp. 415–434, 1988.
14. T. A. Plate. *Distributed Representations and Nested Compositional Structure.* PhD thesis, Department of Computer Science, University of Toronto, 1994.
15. J. Slaney. Scott: A model–guided theorem prover. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 109–114, 1993.
16. P. Smolensky. On the Proper Treatment of Connectionism. *Behavioral and Brain Sciences*, **11**, pp. 1–74, 1988.
17. A. Sperduti. Labeling RAAM. Technical Report TR-93-029, International Computer Science Institute, Berkeley, Ca, 1992.
18. S. Willard. *General Topology.* Addison–Wesley, 1970.

This article was processed using the LaTeX macro package with LLNCS style