Continuity of Semantic Operators in Logic Programming and Their Approximation by Artificial Neural Networks

Pascal Hitzler (1) and Anthony K. Seda (2)

[1] Department of Computer Science, Dresden University, 01062 Dresden, Germany
phitzler@inf.tu-dresden.de, http://www.wv.inf.tu-dresden.de/~pascal
[2] Department of Mathematics, University College Cork, Cork, Ireland
a.seda@ucc.ie, http://euclid.ucc.ie/pages/staff/seda/tseda.htm

Abstract. One approach to integrating first-order logic programming and neural network systems employs the approximation of semantic operators by feedforward networks. For this purpose, it is necessary to view these semantic operators as continuous functions on the reals. This can be accomplished by endowing the space of all interpretations of a logic program with topologies obtained from suitable embeddings. We will present such topologies which arise naturally out of the theory of logic programming, discuss continuity issues of several well-known semantic operators, and derive some results concerning the approximation of these operators by feedforward neural networks.

1 Introduction

The area of neuro-symbolic integration has received growing attention in the recent past. It would be highly desirable to enhance the very adaptable and robust neural networking machinery with the ability to handle structured, symbolic knowledge expressed, for example, as first-order logic programs.

Several attempts to obtain such an integration have been made, and we refer to [1] for a recent survey and to [2] for an overview of the recent challenges in this area. However, most of these attempts have stayed within the context of propositional logic and they are unlikely to carry over to full first-order logic. One exception to this statement is due to Hölldobler, Störr and Kalinke in [3] who employed a general approximation theorem due to Funahashi [4] which states that every continuous function on the real numbers can be uniformly approximated by 3-layer feedforward neural networks. Hölldobler et al. investigated a syntactically restricted class of logic programs (acyclic normal with injective level mapping) and showed that for these a semantic operator (the immediate consequence operator) can, via suitable embeddings, be viewed as a continuous function on the real line. It follows then that the immediate consequence operator can be approximated by neural networks due to Funahashi's theorem. It even turned out in this case that the approximating function is a contraction on the reals, when endowed with the usual metric, and that iterations of the approximating function are well-behaved, in a certain specific sense. We also note that such methods and techniques are quite closely related to the overall aims of the programme of research being undertaken by Blair et al. [5] relating logic programming to continuous models of computation.

In this paper, we will examine the topological content of the results of Hölldobler, Störr and Kalinke, and set out to generalize some of their results. We present our results in two sections, Sections 2 and 3. In the first of these, Section 2, we study continuity issues of semantic operators in logic programming. Then, in Section 3, we relate this discussion to the results of Hölldobler et al. and Funahashi already alluded to.

2 Continuity of Semantic Operators

A (*normal*) *logic program* is a finite set of *clauses* of the form

$$\forall(A \leftarrow L_1 \wedge \cdots \wedge L_n),$$

where $n \in \mathbb{N}$ may differ for each clause, A is an atom in a first order language $\mathcal{L}$ and $L_1, \ldots, L_n$ are literals, that is, atoms or negated atoms, in $\mathcal{L}$. As is customary in logic programming, we will write such a clause in the form

$$A \leftarrow L_1, \ldots, L_n,$$

in which the universal quantifier is understood. Then A is called the *head* of the clause, each L_i is called a *body literal* of the clause and their conjunction $L_1, \ldots, L_n$ is called the *body* of the clause. We allow $n = 0$, by an abuse of notation, which indicates that the body is empty; in this case the clause is called a *unit clause* or a *fact*. We will occasionally use the notation $A \leftarrow \texttt{body}$ for clauses, so that $\texttt{body}$ stands for the conjunction of the body literals of the clause. If no negation symbol occurs in a logic program, the program is called a *definite* logic program. The Herbrand base underlying a given program P will be denoted by B_P and the set of all Herbrand interpretations by I_P, and we note that the latter can be identified simultaneously with the power set of B_P and with the set $\mathbf{2}^{B_P}$ of all functions mapping B_P into the set $\mathbf{2}$ consisting of two distinct elements. By $\mathrm{ground}(P)$, we will denote the set of all ground instances of clauses in P. Finally, we refer the reader to [6] for general background concerning logic programming.

Throughout the rest of the paper, we will impose the standing condition on the language $\mathcal{L}$ that it contains at least one constant symbol and at least one function symbol with arity greater than 0. If this is not done, $\mathrm{ground}(P)$ may be a finite set of ground instances of clauses, and can be treated essentially as a propositional program, for which methods other than those propounded here seem more appropriate.

In logic programming semantics, it has turned out to be both useful and convenient to use many-valued logics. Our investigations will therefore begin by studying suitable topologies on spaces of many-valued interpretations. We assume we have given a finite set $\mathcal{T} = \{t_1, \ldots, t_n\}$ of truth values containing at least the two distinguished values t_1 and t_n, which are interpreted as being the truth values for "false", and "true", respectively. We also assume that we have truth tables for the usual connectives $\vee$, $\wedge$, $\leftarrow$, and $\neg$. Given a logic program P, we denote the set of all (Herbrand) *interpretations* or *valuations* in this logic by $I_{P,n}$; thus $I_{P,n}$ is the set $\mathcal{T}^{B_P}$ of all functions $I : B_P \to \mathcal{T}$. If n is clear from the context, we will use the notation I_P instead of $I_{P,n}$ and we note that this usage is consistent with the one given above for $n = 2$. As usual, any interpretation I can be extended, using the truth tables, to give a truth value in $\mathcal{T}$ to any variable-free formula in $\mathcal{L}$.

Throughout the paper we will make substantial use of elementary notions and results from the mathematical area called *Topology*, our standard reference being [7].

Definition 1. *Given any logic program P, the* generalized atomic topology $\mathcal{Q}$ *on $I_P = I_{P,n}$ is defined to be the product topology on $\mathcal{T}^{B_P}$, where $\mathcal{T} = \{t_1, \ldots, t_n\}$ is endowed with the discrete topology.*

We note that these topologies can be defined analogously for the non-Herbrand case. For $n = 2$, the generalized atomic topology $\mathcal{Q}$ specializes to the query topology of [8] (in the Herbrand case), or to the atomic topology Q of [9] (in the non-Herbrand case). The following results follow immediately since $\mathcal{Q}$ is a product topology of the discrete topology on a finite set, and hence is a topology of pointwise convergence.

Proposition 1. *For $A \in B_P$ and t_i a truth value, let $\mathcal{G}(A, t_i) = \{I \in I_{P,n} \mid I(A) = t_i\}$. The following hold.*

(a) $\mathcal{Q}$ is the topology generated by the subbase $\mathcal{G} = \{\mathcal{G}(A, t_i) \mid A \in B_P, i \in \{1, \ldots, n\}\}$.
(b) A net (I_λ) in I_P converges in $\mathcal{Q}$ if and only if for every $A \in B_P$ there exists some λ_0 such that $I_\lambda(A)$ is constant for all $\lambda \geq \lambda_0$.
(c) $\mathcal{Q}$ is a second countable totally disconnected compact Hausdorff topology which is dense in itself. Hence, $\mathcal{Q}$ is metrizable and homeomorphic to the Cantor topology on the unit interval of the real line.

We note that the second countability of $\mathcal{Q}$ rests on the fact that B_P is countable, so that this property does not in general carry over to the non-Herbrand case.

The study of topologies such as $\mathcal{Q}$ comes from our desire to be able to control the iterative behaviour of semantic operators. Topologies which are closely related to order structures, as common in denotational semantics [10], are of limited applicability since nonmonotonic operators frequently arise naturally in the logic programming context. See also [11, 12] for a study of these issues.

We proceed next with studying a rather general notion of semantic operator, akin to Fitting's approach in [13], which generalizes standard notions occurring in the literature.

Definition 2. *An operator T on I_P is called a* consequence operator *for P if for every $I \in I_P$ the following condition holds: for every ground clause $A \leftarrow \texttt{body}$ in P, where $T(I)(A) = t_i$, say, and $I(\texttt{body}) = t_j$, say, we have that the truth table for $t_i \leftarrow t_j$ yields the truth value t_n, that is, "true".*

It turns out that this notion of consequence operator relates nicely to $\mathcal{Q}$, yielding the following result which was reported by us in [11, 14]. If T is a consequence operator for P and if for any $I \in I_P$ we have that the sequence of iterates $T^m(I)$ converges in $\mathcal{Q}$ to some $M \in I_P$, then M is a model, in a natural sense, for P. Furthermore, continuity of T yields the desirable property that M is a fixed point of T.

Intuitively, consequence operators should propagate "truth" along the implication symbols occurring in the program. From this point of view, we would like the outcome of the truth value of such a propagation to be dependent only on the relevant clause bodies. The next definition captures this intuition.

Definition 3. *Let $A \in B_P$ and denote by $\mathcal{B}_A$ the set of all body atoms of clauses with head A that occur in ground(P). A consequence operator T is called $(P$-$)$*local *if for every $A \in B_P$ and any two interpretations $I, K \in I_P$ which agree on all atoms in $\mathcal{B}_A$, we have $T(I)(A) = T(K)(A)$.*

It is our desire to study continuity in $\mathcal{Q}$ of local consequence operators. Since $\mathcal{Q}$ is a product topology, it is reasonable to expect that finiteness conditions will play a role in this context, and indeed conditions which ensure finiteness in the sense of Definition 4 below, due to [9], have made their appearance in this context.

Definition 4. *Let C be a clause in P and let $A \in B_P$ be such that A coincides with the head of C. The clause C is said to be of* finite type relative to A *if C has only finitely many different ground instances with head A. The program P will be said to be of* finite type relative to A *if each clause in P is of finite type relative to A, that is, if the set of all clauses in ground(P) with head A is finite. Finally, P will be said to be of* finite type *if P is of finite type relative to A for every $A \in B_P$.*

A *local variable* is a variable which appears in a clause body but not in the corresponding head. Local variables appear naturally in practical logic programs, but their occurrence is awkward from the point of view of denotational semantics, especially if they occur in negated body literals since this leads to the so-called floundering problem, see [6].

It is easy to see that, in the context of Herbrand-interpretations, and if function symbols are present, then the absence of local variables is equivalent to a program being of finite type.

Proposition 2. *Let P be a logic program of finite type and let T be a local consequence operator for P. Then T is continuous in $\mathcal{Q}$.*

Proof. Let $I \in I_P$ be an interpretation and let $G_2 = \mathcal{G}(A, t_i)$ be a subbasic neighbourhood of $T(I)$ in $\mathcal{Q}$, and note that G_2 is the set of all $K \in I_P$ such that $K(A) = t_i$. We need to find a neighbourhood G_1 of I such that $T(G_1) \subseteq G_2$. Since P is of finite type, the set $\mathcal{B}_A$ is finite. Hence the set $G_1 = \bigcap_{B \in \mathcal{B}_A} \mathcal{G}(B, I(B))$ is a finite intersection of open sets and is therefore open. Since each $K \in G_1$ agrees with I on $\mathcal{B}_A$, we obtain $T(K)(A) = T(I)(A) = t_i$ for each $K \in G_1$ by locality of T. Hence, $T(G_1) \subseteq G_2$.

Now, if P is not of finite type, but we can ensure by some other property of P that the possibly infinite intersection $\bigcap_{B \in \mathcal{B}_A} \mathcal{G}(B, I(B))$ is open, then the above proof will carry over to programs which are not of finite type. Alternatively, we would like to be able to disregard the infinite intersection entirely under conditions which ensure that we have to consider finite intersections only, as in the case of a program of finite type. The following definition is, therefore, quite a natural one to make.

Definition 5. *Let P be a logic program and let T be a consequence operator on I_P. We say that T is (P-)*locally finite *for $A \in B_P$* and *$I \in I_P$ if there exists a finite subset $S = S(A, I) \subseteq \mathcal{B}_A$ such that we have $T(J)(A) = T(I)(A)$ for all $J \in I_P$ which agree with I on S. We say that T is (P-)*locally finite *if it is locally finite for all $A \in B_P$ and all $I \in I_P$.*

It is easy to see that a locally finite consequence operator is local. Conversely, a local consequence operator for a program of finite type is locally finite. This follows from the observation that for a program of finite type the sets $\mathcal{B}_A$, for any $A \in B_P$, are finite. But a much stronger result holds.

Theorem 1. *A local consequence operator is locally finite if and only if it is continuous in $\mathcal{Q}$.*

Proof. Let T be a locally finite consequence operator, let $I \in I_P$, let $A \in B_P$, and let $G_2 = \mathcal{G}(A, T(I)(A))$ be a subbasic neighbourhood of $T(I)$ in $\mathcal{Q}$. Since T is locally finite, there is a finite set $S \subseteq \mathcal{B}_A$ such that $T(J)(A) = T(I)(A)$ for all $J \in \bigcap_{B \in S} \mathcal{G}(B, I(B))$. By finiteness of S, the set $\bigcap_{B \in S} \mathcal{G}(B, I(B))$ is open, which suffices for continuity of T.

For the converse, assume that T is continuous in $\mathcal{Q}$ and let $A \in B_P$ and $I \in I_P$ be chosen arbitrarily. Then $G_2 = \mathcal{G}(A, T(I)(A))$ is a subbasic open set, so that, by continuity of T, there exists a basic open set $G_1 = \mathcal{G}(B_1, I(B_1)) \cap \cdots \cap \mathcal{G}(B_k, I(B_k))$ with $T(G_1) \subseteq G_2$. In other words, we have $T(J)(A) = T(I)(A)$ for each $J \in \bigcap_{B \in S'} \mathcal{G}(B, I(B))$, where $S' = \{B_1, \dots, B_k\}$ is a finite set. Since T is local, the value of $T(J)(A)$ depends only on the values $J(A)$ of atoms $A \in \mathcal{B}_A$. So if we set $S = S' \cap \mathcal{B}_A$, then $T(J)(A) = T(I)(A)$ for all $J \in \bigcap_{B \in S} \mathcal{G}(B, I(B))$ which is to say that T is locally finite for A and I. Since A and I were chosen arbitrarily, we obtain that T is locally finite.

The following corollary was communicated to us by Howard A. Blair in the two-valued case. A *level mapping* for a program P is a mapping $l : B_P \to \alpha$ for some ordinal α; we always assume that l has been extended to all literals by setting $l(\neg A) = l(A)$ for each $A \in B_P$. An *ω-level mapping* for P is a level mapping $l : B_P \to \mathbb{N}$.

Corollary 1. *Let P be a program, let T be a local consequence operator and let l be an injective ω-level mapping for P with the following property: for each $A \in B_P$ there exists an $n_A \in \mathbb{N}$ such that $l(B) < n_A$ for all $B \in \mathcal{B}_A$. Then T is continuous in $\mathcal{Q}$.*

Proof. It follows easily from the given conditions that $\mathcal{B}_A$ is finite for all $A \in B_P$, which implies that T is locally finite.

We next take a short detour from our discussion of continuity to study the weaker notion of measurability [15] for consequence operators. For a collection M of subsets of a set X, we denote by $\sigma(M)$ the smallest σ-algebra containing M, called the σ-algebra *generated by* M. Recall that a function $f : X \to X$ is measurable with respect to $\sigma(M)$ if and only if $f^{-1}(A) \in \sigma(M)$ for each $A \in M$. If β is the subbase of a topology τ and β is countable, then $\sigma(\beta) = \sigma(\tau)$. It turns out that local consequence operators are always measurable with respect to the σ-algebra generated by a generalized atomic topology. The following result generalizes a theorem from [16].

Theorem 2. *Local consequence operators are measurable with respect to $\sigma(\mathcal{G}) = \sigma(\mathcal{Q})$.*

Proof. Let T be a local consequence operator. We need to show that for each subbasic set $\mathcal{G}(A, t_i)$ we have $T^{-1}(\mathcal{G}(A, t_i)) \in \sigma(\mathcal{G})$.

Let $A \in B_P$ and let $t \in \mathcal{T}$ both be chosen arbitrarily. Let F be the set of all functions from $\mathcal{B}_A$ to $\mathcal{T}$, and note that F is countable since $\mathcal{B}_A$ is countable and $\mathcal{T}$ is finite. Let F' be the subset of F which contains all functions f with the following property: whenever an interpretation I agrees with f on $\mathcal{B}_A$, then $T(I)(A) = t$. Then, $\bigcap_{B \in \mathcal{B}_A} \mathcal{G}(B, f(B)) \in T^{-1}(\mathcal{G}(A, t))$ for each $f \in F'$.

We obtain by locality of T, that whenever I is an interpretation for which $T(I)(A) = t$ holds, then there exists a function $f_I \in F'$ such that f_I and I agree on $\mathcal{B}_A$, and this yields $T^{-1}(\mathcal{G}(A, t)) = \bigcup_{f_I \in F'} \bigcap_{B \in \mathcal{B}_A} \mathcal{G}(B, I(B))$. Since F' and $\mathcal{B}_A$ are countable, the set on the right hand side is measurable as required.

We turn now to the study of the continuity of a particular operator introduced by Fitting [13] to logic programming semantics. To this end, we associate a set P^* with each logic program P by the following construction. Let $A \in B_P$. If A occurs as the head of some unit clause $A \leftarrow$ in ground(P), then replace it by the clause $A \leftarrow t_n$, where by a slight abuse of notation we interpret t_n to be an additional atom which we adjoin to the language $\mathcal{L}$ and always evaluate to $t_n \in \mathcal{T}$, that is, it evaluates to "true". If A does not occur in the head of any clause in ground(P), then add the clause $A \leftarrow t_0$, where t_0 is interpreted as an additional atom which again we adjoin to $\mathcal{L}$ and always evaluate to $t_0 \in \mathcal{T}$, that is, it evaluates to "false". The resulting (ground) program, which results from ground(P) by the changes just given with respect to every $A \in B_P$, will be denoted by P'. Now let P^* be the set of all *pseudo clauses* of the form

$A \leftarrow C_1 \vee C_2 \vee \ldots$, where the C_i are exactly the bodies of the clauses in P' with head A. We call A the *head* and $B_A = C_1 \vee C_2 \vee \ldots$ the *body* of the resulting pseudo clause, and we note that each $A \in B_P$ occurs in the head of exactly one pseudo clause in P^*. Bodies of pseudo clauses are possibly infinite disjunctions, but this will not pose any particular difficulty with respect to the logics which we are going to discuss. We note that a program P is of finite type if and only if all bodies of all pseudo clauses in P^* are finite.

Now, if we are given (suitable) truth tables for negation, conjunction and disjunction, we are able to evaluate the truth values of bodies of pseudo clauses relative to given interpretations.

Definition 6. *Let P be a logic program. Define the mapping $F_P : I_{P,n} \to I_{P,n}$ relative to a given (suitable) logic with n truth values by $F_P(I) = J$, where J assigns to each $A \in B_P$ the truth value $I(B_A)$.*

We call operators which satisfy Definition 6 *Fitting operators*. If we impose the mild assumption that $t_j \leftarrow t_j$ evaluates to "true" for every j with respect to the underlying logic, then we easily obtain that every Fitting operator is a local consequence operator. This will always be the case in the remaining section.

The virtue of Definition 6, due to Fitting [13], lies in the fact that several operators known from the theory of logic programming can be derived from it in a very concise way, and we refer to [13, 17] for a discussion of these matters, see also [14]. We will now investigate some of these operators in the light of Theorem 1. In the following, we will denote the "true" truth value by $\mathbf{t}$ and the "false" truth value by $\mathbf{f}$.

If the chosen logic is classical two-valued logic, then the corresponding Fitting operator is the *single-step* or *immediate consequence operator* T_P (for a given program P). Now, if $T_P(I)(A) = \mathbf{t}$, then there exists a clause $A \leftarrow \texttt{body}$ in ground(P) such that $I(\texttt{body})$ is true, and we obtain $T_P(J)(A) = \mathbf{t}$ whenever $J(\texttt{body}) = \mathbf{t}$. The observation that bodies of clauses are finite conjunctions leads us to conclude the following lemma.

Lemma 1. *If $T_P(I)(A)$ is true, then T_P is locally finite for A and I. Furthermore, T_P is continuous if and only if it is locally finite for all A and I with $T_P(I)(A) = \mathbf{f}$.*

A body $\bigvee C_i$ of a pseudo clause is false if and only if all C_i are false. Since T_P is a Fitting operator, we obtain $T_P(I)(A) = \mathbf{f}$ if and only if all C_i are false. If we require T_P to be locally finite for A and I, then there must be a finite set $S \subseteq \mathcal{B}_A$ such that any $J \in I_P$ which agrees with I on S renders all C_i to be false. These observations now easily yield the following theorem from [9].

Theorem 3. *Let P be a normal logic program. Then T_P is continuous if and only if, for each $I \in I_P$ and for each $A \in B_P$ with $T_P(I)(A) = \mathbf{f}$, either there is no clause in P with head A or there exists a finite set $S(I, A) = \{A_1, \ldots, A_k, B_1, \ldots, B_{k'}\} \subseteq \mathcal{B}_A$ with the following properties:*

(i) $A_1, \ldots, A_k$ are true in I and $B_1, \ldots, B_{k'}$ are false in I.

(ii) Given any clause C with head A, at least one $\neg A_i$ or at least one B_j occurs in the body of C.

Table 1. Connectives for Kleene's strong three-valued logic.

p	q	$p \wedge q$	$p \vee q$	$\neg p$
t	t	t	t	f
t	u	u	t	f
t	f	f	t	f
u	t	u	t	u
u	u	u	u	u
u	f	f	u	u
f	t	f	t	t
f	u	f	u	t
f	f	f	f	t

In the case of Kleene's strong three-valued logic, with set of truth values $\mathcal{T} = \{t, u, f\}$ and logical connectives as in Table 1, the associated Fitting operator was introduced in [18] and is denoted by Φ_P, for a given program P. As in the case of classical two-valued logic, we obtain the following lemma.

Lemma 2. *If $\Phi_P(I)(A) = t$, then Φ_P is locally finite for A and I. Furthermore, Φ_P is continuous if and only if it is locally finite for all A and I with $\Phi_P(I)(A) \in \{u, f\}$.*

Obtaining a theorem analogous to Theorem 3 is now straightforward, but tedious, and we omit the details. Similar considerations apply to the operator Ψ on Belnap's four-valued logic [13] and to the operators from [19].

We mention in passing the nonmonotonic Gelfond-Lifschitz operator [20] in classical two-valued logic, whose fixed points yield the stable models of the program in question. It turns out that this operator is not a consequence operator in the sense discussed in this paper, and attempts to characterize continuity of it will involve different methods, e.g. by means of the results from [21].

3 Approximation by Artificial Neural Networks

A *3-layer feedforward network* (or *single hidden layer feedforward network*) consists of an *input layer*, a *hidden layer*, and an *output layer*. Each layer consists of finitely many *computational units*. There are connections from units in the input layer to units in the hidden layer, and from units in the hidden layer to units in the output layer. The input-output relationship of each unit is represented by *inputs* x_i, *output* y, *connection weights* w_i, *threshold* θ, and a function ϕ as follows:

$$y = \phi\left(\sum_i w_i x_i - \theta\right).$$

The function ϕ, which we will call the *squashing function* of the network, is usually non-constant, bounded and monotone increasing, and sometimes also assumed to be continuous. We will specify the requirements on ϕ that we assume in each case.

We assume throughout that the input-output relationships of the units in the input and output layer are linear. The output function of a network as described above is then obtained as a mapping $f : \mathbb{R}^r \to \mathbb{R}$ with

$$f(x_1, \ldots, x_r) = \sum_j c_j \phi \left(\sum_i w_{ji} x_i - \theta_j \right),$$

where r is the number of units in the input layer, and the constants c_j correspond to weights from hidden to output layers. See also Figure 1. We refer to [22] for background concerning artificial neural networks.

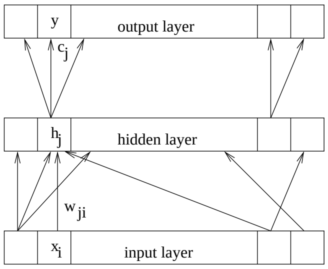

Fig. 1. 3-Layer Feedforward Neural Network.

It is our aim to obtain results on the approximation of consequence operators by input-output functions of 3-layer feedforward networks. Our first result rests on the following theorem, which is due to Funahashi, see [4].

Theorem 4. *Suppose that $\phi : \mathbb{R} \to \mathbb{R}$ is non-constant, bounded, monotone increasing and continuous. Let $K \subseteq \mathbb{R}^n$ be compact, let $f : K \to \mathbb{R}$ be a continuous mapping and let $\varepsilon > 0$. Then there exists a 3-layer feedforward network with squashing function ϕ whose input-output mapping $\bar{f} : K \to \mathbb{R}$ satisfies $\max_{x \in K} d(f(x), \bar{f}(x)) < \varepsilon$, where d is a metric which induces the natural topology on $\mathbb{R}$.*

In other words, each continuous function $f : K \to \mathbb{R}$ can be uniformly approximated by input-output functions of 3-layer networks. For our purposes, it will suffice to assume that K is a compact subset of the set of real numbers, so that our network architecture can be depicted as in Figure 2.

The Cantor set $\mathcal{C}$ is a compact subset of the real line and the topology which $\mathcal{C}$ inherits as a subspace of $\mathbb{R}$ coincides with the Cantor topology on $\mathcal{C}$. Also,

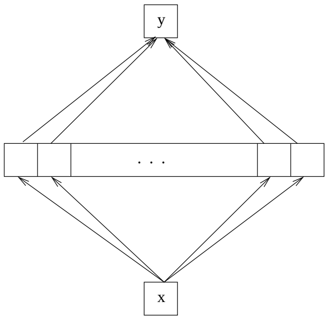

Fig. 2. Network architecture used in this paper.

the Cantor space $\mathcal{C}$ is homeomorphic to $I_{P,n}$ when the latter is endowed with a generalized atomic topology $\mathcal{Q}$. Hence, if a consequence operator T is continuous in $\mathcal{Q}$, we can identify it with a mapping $\iota(T) : \mathcal{C} \to \mathcal{C} : x \mapsto \iota(T(\iota^{-1}(x)))$ which is continuous in the subspace topology of $\mathcal{C}$ in $\mathbb{R}$, as follows.

Theorem 5. *Let P be a program, let T be a consequence operator which is locally finite and let ι be a homeomorphism from $(I_{P,n}, \mathcal{Q})$ to $\mathcal{C}$. Then T (more precisely $\iota(T)$) can be uniformly approximated by input-output mappings of 3-layer feedforward networks.*

Proof. Under the conditions stated in the theorem, the operator T is continuous in $\mathcal{Q}$. Using the homeomorphism ι, the resulting function $\iota(T)$ is continuous on the Cantor set $\mathcal{C}$, which is a compact subset of $\mathbb{R}$. Applying Theorem 4, $\iota(T)$ can be uniformly approximated by input-output functions of 3-layer feedforward networks.

The restriction to programs with continuous consequence operator is unsatisfactory. There is another approximation theorem due to [23], which requires only measurable functions:

Theorem 6. *Suppose that ϕ is a monotone increasing function from $\mathbb{R}$ onto $(0, 1)$. Let $f : \mathbb{R}^r \to \mathbb{R}$ be a Borel-measurable function and let μ be a probability Borel-measure on $\mathbb{R}^r$. Then, given any $\varepsilon > 0$, there exists a 3-layer feedforward network with squashing function ϕ whose input-output function $\bar{f} : \mathbb{R}^r \to \mathbb{R}$ satisfies*

$$\varrho_\mu(f, \bar{f}) = \inf\{\delta > 0 : \mu\{x : |f(x) - \bar{f}(x)| > \delta\} < \delta\} < \varepsilon.$$

In other words, the class of functions computed by 3-layer feedforward neural nets is dense in the set of all Borel measurable functions $f : \mathbb{R}^r \to \mathbb{R}$ relative to the metric ϱ_μ defined in Theorem 6.

By means of Theorem 2, we can now view a local consequence operator T as a measurable function $\iota(T)$ on $\mathcal{C}$ by identifying $I_{P,n}$ with $\mathcal{C}$ via a homeomorphism ι.

Since $\mathcal{C}$ is measurable as a subset of the real line, this operator can be extended[3] to a measurable function on $\mathbb{R}$ and we obtain the following result.

Theorem 7. *Given any program P with local consequence operator T, the operator T (more precisely $\iota(T)$) can be approximated in the manner of Theorem 6 by input-output mappings of 3-layer feedforward networks.*

This result is somewhat unsatisfactory since the approximation stated in Theorem 6 is only *almost everywhere*, that is, pointwise with the exception of a set of measure zero. The Cantor set, however, is a set of measure zero. In [16], the present authors were able to strengthen this result in the case of classical logic by giving an explicit extension of T to the real line.

We want to return now to the case discussed earlier in Theorem 5. In [3], the following recurrent neural network architecture was considered: we assume that the number of output and input units is equal and that, after each propagation through the network, the output values are fed back without changes into input values. For the case which we consider it will again be sufficient to suppose that the input layer consists of one unit only, so that the architecture can be depicted as in Figure 3.

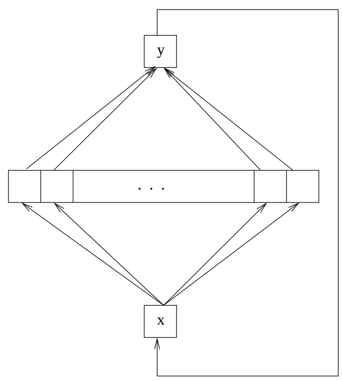

Fig. 3. Recurrent Network.

We will show in the following that iterates of locally finite local consequence operators can be approximated arbitrarily closely by iterates of suitably chosen networks. This is in fact a consequence of the uniform approximation obtained from Theorem 4 and the compactness of the unit interval.

[3] E.g. as a function $T : \mathbb{R} \to \mathbb{R}$ with $T(x) = \iota(T_P(\iota^{-1}(x)))$ if $x \in \mathcal{C}$ and $T(x) = 0$ otherwise.

Let P be a logic program, let T be a locally finite local consequence operator for P and let $\iota : I_P \to \mathcal{C}$ be a homeomorphism. Let F be a continuous extension of $\iota(T)$ onto the unit interval $[0,1]$ in the reals, let d be the natural metric on $\mathbb{R}$, and let $\varepsilon > 0$. By Theorem 5, there exists a three-layer feedforward network with input-output mapping f such that $\max_{x\in[0,1]} d(f(x), F(x)) < \varepsilon$. Since $[0,1]$ is compact, and F is continuous, we obtain that F is Lipschitz-continuous, that is, there exists $\lambda \geq 0$ such that for all $x, y \in [0,1]$ we have $d(F(x), F(y)) \leq \lambda d(x,y)$. For $x, y \in [0,1]$ we therefore obtain

$$d(f(x), F(y)) \leq d(f(x), F(x)) + d(F(x), F(y)) \leq \varepsilon + \lambda d(x,y). \tag{1}$$

Now let $x \in [0,1]$ be arbitrarily chosen. By Equation (1) we obtain

$$d(f^2(x), F^2(x)) \leq \varepsilon + \lambda d(f(x), F(x)) \leq \varepsilon + \lambda\varepsilon. \tag{2}$$

Inductively, we can prove that for all $n \in \mathbb{N}$ we have

$$d(f^n(x), F^n(x)) \leq \varepsilon + \lambda\varepsilon + \cdots + \lambda^{n-1}\varepsilon = \varepsilon\left(\sum_{i=0}^{n-1} \lambda^i\right) = \varepsilon\frac{1-\lambda^n}{1-\lambda}. \tag{3}$$

Thus we obtain the following bound on the error produced by the recurrent network after n iterations.

Theorem 8. *With the notation and hypotheses above, for any $I \in I_P$ and any $n \in \mathbb{N}$ we have*

$$|f^n(\iota(I)) - \iota(T^n(I))| \leq \varepsilon\frac{1-\lambda^n}{1-\lambda}.$$

Proof. Note that $\iota(T^n(I)) = F^n(\iota(I))$, and the assertion follows from Equation (3) since d is the natural metric on $\mathbb{R}$.

We derive a few corollaries from this result.

Corollary 2. *If F is a contraction on $[0,1]$, so that $\lambda < 1$, then $(F^k(\iota(I)))$ converges for every I to the unique fixed point x of F and there exists $m \in \mathbb{N}$ such that for all $n \geq m$ we have*

$$|f^n(\iota(I)) - x| \leq \varepsilon\frac{1}{1-\lambda}.$$

Proof. The convergence follows from the Banach contraction mapping theorem. The inequality follows immediately from Theorem 8 using the well-known equation for limits of geometric series.

If F is a contraction on $[0,1]$, then T is a contraction on the complete subspace $\mathcal{C}$, and also has a fixed point M, and $\iota(M) = x$. However, it seems difficult to guarantee the hypothesis of Corollary 2, although Hölldobler et al. have achieved in [3] a similar result for acyclic programs with injective level mappings in classical logic. The following result may be more promising.

Corollary 3. *If for some $I \in I_P$, $T^n(I)$ converges in $\mathcal{Q}$ to a fixed point M of T, then for every $\delta > 0$ there exists a network with input-output function f, and some $n \in \mathbb{N}$ such that $|f^n(\iota(I)) - \iota(M)| < \delta$.*

Proof. The hypothesis implies that $F^n(\iota(I))$ converges to $\iota(M)$ in the natural metric on $\mathbb{R}$. Given $\delta > 0$, there exists $n \in \mathbb{N}$ such that $|F^m(\iota(I)) - \iota(M)| < \frac{\delta}{2}$ for all $m \geq n$. Since F is fixed, we know the value of λ. Now, by the approximation results above, we choose a network with input-output function f such that $\varepsilon \frac{1-\lambda^n}{1-\lambda} < \frac{\delta}{2}$. Then using Theorem 8 and the triangle inequality we obtain

$$\begin{aligned} |f^n(\iota(I)) - \iota(M)| &\leq |f^n(\iota(I)) - F^n(\iota(I))| + |F^n(\iota(I)) - \iota(M)| \\ &< 2 \cdot \frac{\delta}{2} \leq \delta. \end{aligned}$$

We will close by describing a class of programs for which the additional hypothesis from Corollary 3 is satisfied. The result is well-known for the case of classical two-valued logic and the immediate consequence operator in this case. The following definition is due to [24].

Definition 7. *A logic program P is called* acyclic *if there exists an ω-level mapping l such that for each clause $A \leftarrow L_1, \ldots, L_n$ in ground(P) we have $l(A) > l(L_i)$ for all $i = 1, \ldots, n$.*

In the following, let P be acyclic with level mapping l, and let T be a local consequence operator for P. We next define a mapping $d : I_P \times I_P \to \mathbb{R}$ by $d(I, J) = 2^{-n}$, where n is least such that I and J differ on some atom A with $l(A) = n$. It is easily verified that d is a complete metric on I_P, see [25].

Proposition 3. *With the stated hypotheses, T is a contraction with respect to d.*

Proof. Suppose $d(I, J) = 2^{-n}$. Then I and J coincide on all atoms of level less than n. Now let $A \in B_P$ with $l(A) = n$. Then by acyclicity of P we have that all atoms in $\mathcal{B}_A$ are of level less than n, and by locality of T we have that $T(I)(A) = T(J)(A)$. So $d(T(I), T(J)) \leq 2^{-(n+1)}$.

We finally obtain the following theorem.

Theorem 9. *Let P be an acyclic program and let T be a local consequence operator for P. Then for any $I \in I_P$ we have that $T^n(I)$ converges in $\mathcal{Q}$ to the unique fixed point M of T.*

Proof. By Proposition 3, and since d is a complete metric, we can apply the Banach contraction mapping theorem which yields convergence of $T^n(I)$ in d to a unique fixed point M of T. By definition of d, the convergence of the sequence of valuations $T^n(I)$ to M must be pointwise, hence is also convergence in $\mathcal{Q}$.

Theorem 9 is remarkable since the existence of a fixed point of the semantic operator can be guaranteed without any further knowledge about the underlying multi-valued logic.

4 Conclusions

The contribution of this paper is twofold.

(1) We have motivated the study of local consequence operators in multi-valued logic from an abstract and topological point of view. So far, semantic operators in multi-valued logic have been monotonic, to our knowledge, in order to circumvent problems related to nonmonotonicity of semantic operators in the study of declarative semantics. However, it is becoming clearer and clearer that nonmonotonic semantic operators in logic programming can be controlled using topological methods, see [11, 12]. By merging multi-valued logics and nonmonotonicity the arsenal of tools available to researchers working on the declarative semantics of logic programming becomes considerably larger.

(2) We have generalized substantially some results from [3] concerning the approximation of semantic operators by artificial neural networks. However, many questions remain. We have not yet found a way to actually construct the approximating network. Very recent results [26] indicate that in order to do this, it will be necessary to find ways of obtaining good bounds on Lipschitz constants of semantic operators. Also, alternative methods of extending the results for continuous operators are needed, since the approach based on measurability seems unsatisfactory. One may get better results by using different homeomorphisms or even by using p-adic numbers[4] for representing I_P as a subspace of $\mathbb{R}$.

Acknowledgement. We would like to thank Howard A. Blair, Steffen Hölldobler, and Hans-Peter Störr for discussions on the subject matter.

References

1. Browne, A., Sun, R.: Connectionist inference models. Neural Networks **14** (2001) 1331–1355
2. Hölldobler, S.: Challenge problems for the integration of logic and connectionist systems. In Bry, F., Geske, U., Seipel, D., eds.: Proceedings 14. Workshop Logische Programmierung. Volume 90 of GMD Report., GMD (2000) 161–171
3. Hölldobler, S., Störr, H.P., Kalinke, Y.: Approximating the semantics of logic programs by recurrent neural networks. Applied Intelligence **11** (1999) 45–58
4. Funahashi, K.I.: On the approximate realization of continuous mappings by neural networks. Neural Networks **2** (1989) 183–192
5. Blair, H.A., Dushin, F., Jakel, D.W., Rivera, A.J., Sezgin, M.: Continuous models of computation for logic programs. In Apt, K.R., Marek, V.W., Truszczyński, M., Warren, D.S., eds.: The Logic Programming Paradigm: A 25-Year Persepective. Springer, Berlin (1999) 231–255
6. Lloyd, J.W.: Foundations of Logic Programming. Springer, Berlin (1988)
7. Willard, S.: General Topology. Addison-Wesley, Reading, MA (1970)
8. Batarekh, A., Subrahmanian, V.: Topological model set deformations in logic programming. Fundamenta Informaticae **12** (1989) 357–400
9. Seda, A.K.: Topology and the semantics of logic programs. Fundamenta Informaticae **24** (1995) 359–386

[4] Suggested by Howard A. Blair.

10. Abramsky, S., Jung, A.: Domain theory. In Abramsky, S., Gabbay, D., Maibaum, T.S., eds.: Handbook of Logic in Computer Science. Volume 3. Clarendon, Oxford (1994)
11. Hitzler, P.: Generalized Metrics and Topology in Logic Programming Semantics. PhD thesis, Department of Mathematics, National University of Ireland, University College Cork (2001)
12. Hitzler, P., Seda, A.K.: Generalized metrics and uniquely determined logic programs. Theoretical Computer Science (200x) to appear
13. Fitting, M.: Fixpoint semantics for logic programming — A survey. Theoretical Computer Science **278** (2002) 25–51
14. Hitzler, P., Seda, A.K.: Semantic operators and fixed-point theory in logic programming. In: Proceedings of the joint IIIS & IEEE meeting of the 5th World Multiconference on Systemics, Cybernetics and Informatics, SCI2001 and the 7th International Conference on Information Systems Analysis and Synthesis, ISAS2001, Orlando, Florida, USA, International Institute of Informatics and Systemics: IIIS (2001)
15. Bartle, R.G.: The Elements of Integration. John Wiley & Sons, New York (1966)
16. Hitzler, P., Seda, A.K.: A note on relationships between logic programs and neural networks. In Gibson, P., Sinclair, D., eds.: Proceedings of the Fourth Irish Workshop on Formal Methods, IWFM'00. Electronic Workshops in Comupting (eWiC), British Computer Society (2000)
17. Denecker, M., Marek, V.W., Truszczynski, M.: Approximating operators, stable operators, well-founded fixpoints and applications in non-monotonic reasoning. In Minker, J., ed.: Logic-based Artificial Intelligence. Kluwer Academic Publishers, Boston (2000) 127–144
18. Fitting, M.: A Kripke-Kleene-semantics for general logic programs. The Journal of Logic Programming **2** (1985) 295–312
19. Hitzler, P., Seda, A.K.: Characterizations of classes of programs by three-valued operators. In Gelfond, M., Leone, N., Pfeifer, G., eds.: Logic Programming and Nonmonotonic Reasoning, Proceedings of the 5th International Conference on Logic Programming and Non-Monotonic Reasoning, LPNMR'99, El Paso, Texas, USA. Volume 1730 of Lecture Notes in Artificial Intelligence., Springer, Berlin (1999) 357–371
20. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R.A., Bowen, K.A., eds.: Logic Programming. Proceedings of the 5th International Conference and Symposium on Logic Programming, MIT Press (1988) 1070–1080
21. Wendt, M.: Unfolding the well-founded semantics. Journal of Electrical Engineering, Slovak Academy of Sciences **53** (2002) 56–59 (Proceedings of the 4th Slovakian Student Conference in Applied Mathematics, Bratislava, April 2002).
22. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
23. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2** (1989) 359–366
24. Cavedon, L.: Acyclic programs and the completeness of SLDNF-resolution. Theoretical Computer Science **86** (1991) 81–92
25. Fitting, M.: Metric methods: Three examples and a theorem. The Journal of Logic Programming **21** (1994) 113–127
26. Bader, S.: From logic programs to iterated function systems. Master's thesis, Department of Computer Science, Dresden University of Technology (2003)