3-2004

Corollaries on the Fixpoint Completion: Studying the Stable Semantics by Means of the Clark Completion

Pascal Hitzler
pascal.hitzler@wright.edu

Follow this and additional works at: https://corescholar.libraries.wright.edu/cse

Part of the Computer Sciences Commons, and the Engineering Commons

Repository Citation

This Conference Proceeding is brought to you for free and open access by Wright State University's CORE Scholar. It has been accepted for inclusion in Computer Science and Engineering Faculty Publications by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu, library-core scholar@wright.edu.
Corollaries on the fixpoint completion: studying the stable semantics by means of the Clark completion

Pascal Hitzler**

Department of Computer Science, Dresden University of Technology
www.wv.inf.tu-dresden.de/~pascal/
phitzler@inf.tu-dresden.de

Abstract. The fixpoint completion fix(P) of a normal logic program P is a program transformation such that the stable models of P are exactly the models of the Clark completion of fix(P). This is well-known and was studied by Dung and Kanchanasut [15]. The correspondence, however, goes much further: The Gelfond-Lifschitz operator of P coincides with the immediate consequence operator of fix(P), as shown by Wendt [51], and even carries over to standard operators used for characterizing the well-founded and the Kripke-Kleene semantics. We will apply this knowledge to the study of the stable semantics, and this will allow us to almost effortlessly derive new results concerning fixed-point and metric-based semantics, and neural-symbolic integration.

1 Introduction

The fixpoint completion of normal logic programs was introduced in [15], and independently under the notion of residual program in [9]. In essence, the fixpoint completion fix(P) of a given program P is obtained by performing a complete unfolding through all positive body literals in the program, and by disregarding all clauses with remaining positive body literals. Its importance lies in the fact that the stable models [20] of P are exactly the supported models of fix(P), i.e. the models of the Clark completion [11] of fix(P). Also, the well-founded model [50] of P is exactly the Fitting or Kripke-Kleene model [16] of fix(P). These correspondences are well-known and have been employed by many authors for investigating the stable and the well-founded semantics, see e.g. [7].

The relation between a program and its fixpoint completion, however, is not exhausted by the correspondences between the different semantics just mentioned: It also concerns the semantic operators underlying these semantics, as shown in [51]. The virtue of this observation lies in the fact that it allows to carry over operator-based results on the supported, respectively, Fitting semantics, to

** This work was supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD) and carried out while the author was visiting the Department of Electrical Engineering and Computer Science at Case Western Reserve University, Cleveland, Ohio.
the stable, respectively, well-founded semantics. To the best of our knowledge, this has not been noted before.

In this paper, we display the strength of the operator-based correspondence by drawing a number of corollaries on the stable semantics from it. While these results are of interest in their own right, they do not constitute the main point we want to make here. Some of them are not even new, although we give new proofs. The goal of this paper is to provide a new technical tool for studying the stable and the well-founded semantics, namely the correspondences via the fixpoint completion between the semantic operators mentioned. To display this, we draw several corollaries from results in the literature, which are all valid for logic programs over a first-order language.

The structure of the paper is as follows. In Section 2 we recall the fixpoint completion and the results due to [51] which provide the starting points for our report. In Section 3 we study continuity of the Gelfond-Lifschitz operator in the Cantor topology, thereby providing technical results which will be of use later. In Section 4 we study methods for obtaining stable models by means of limits of iterates of the Gelfond-Lifschitz operator, and in Section 5 we will discuss results on the representation of logic programs by artificial neural networks. We briefly conclude in Section 6.

Acknowledgement. Thanks go to Matthias Wendt for helpful discussions and comments.

2 The Fixpoint Completion

A (normal) logic program is a finite set of universally quantified clauses of the form
\[ \forall(A \leftarrow L_1 \land \cdots \land L_n), \]
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, or even as
\[ A : -L_1, \ldots, L_n \]
following Prolog notation. Then \( A \) is called the head of the clause, each \( L_i \) is called a body literal of the clause and their conjunction \( L_1 \cdots L_n \) 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. If no negation symbol occurs in a logic program, the program is called a definite logic program. The Herbrand base underlying a given program \( P \), i.e. the set of all ground instances of atoms over \( \mathcal{L} \), 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 \( 2^{B_P} \) of all functions mapping \( B_P \) into the set \( 2 \) consisting of two distinct elements. Since the set \( I_P \) is the power set of \( B_P \), it carries set-inclusion as natural ordering, which makes it a complete lattice. By \( \text{ground}(P) \) we denote the (possibly infinite) set of all ground instances of clauses in \( P \).

The single-step or immediate consequence operator [37] of \( P \) is defined as a function \( T_P : I_P \rightarrow I_P \), where \( T_P(I) \) is the set of all \( A \in B_P \) for which there exists a clause \( \neg B_1 \ldots \neg B_m \) in \( \text{ground}(P) \), with \( B_1 \ldots B_m \in I \). This \( T_P \) is a fixed point of \( T_P \). Supported models correspond to models of the Clark completion of \( P \), as noted in [1]. The pre-fixed points of \( T_P \), i.e. interpretations \( I \in I_P \) with \( I \subseteq T_P(I) \), are exactly the Herbrand models of \( P \), in the sense of first-order logic. If \( P \) is definite, then \( T_P \) is in fact a Scott- (or order-) continuous operator on \( I_P \) [37], and its least fixed point \( \text{fix}(T_P) \) coincides with the least Herbrand model of \( P \). The least fixed point, in this case, can be obtained as \( \text{fix}(T_P) = T_P \uparrow \omega := \cup_n T_P \uparrow n \), where \( T_P \uparrow 0 = \emptyset \) and recursively \( T_P \uparrow (n+1) = T_P(T_P \uparrow n) \).

The Gelfond-Lifschitz transformation [20] of a program \( P \) with respect to an interpretation \( I \) is denoted by \( P/I \), and consists of exactly those clauses \( A \quad A_1 \ldots A_n \quad \neg B_1 \ldots \neg B_m \) in \( \text{ground}(P) \), with \( A_1 \ldots A_n \in I \). We call \( I \in I_P \) a stable model of \( P \) if it is a fixed point of \( \text{GLP} \).

**Definition 1.** A quasi-interpretation\(^1\) is a set of clauses of the form \( A \quad \neg B_1 \ldots \neg B_m \), where \( A \) and \( B_i \) are ground atoms for all \( i = 1 \ldots m \). Given a normal logic program \( P \) and a quasi-interpretation \( Q \), we define \( T_P(Q) \) to be the quasi-interpretation consisting of the set of all clauses

\[
A \quad \text{body}_1 \ldots \text{body}_n \quad \neg B_1 \ldots \neg B_m
\]

for which there exists a clause

\[
A \quad A_1 \ldots A_n \quad \neg B_1 \ldots \neg B_m
\]

in \( \text{ground}(P) \) and clauses \( A_i \text{ body}_i \) in \( Q \) for all \( i = 1 \ldots n \). We explicitly allow the cases \( n = 0 \) or \( m = 0 \) in this definition.

Note that the set of all quasi-interpretations is a complete partial order (cpo) with respect to set-inclusion. It was shown in [15], that for normal programs \( P \), the operator \( T_P \) is Scott-continuous on the set of all quasi-interpretations. So we can define the fixpoint completion \( \text{fix}(P) \) of \( P \) by \( \text{fix}(P) = T_P \uparrow \omega \), i.e. \( \text{fix}(P) \) is the least fixed point of the operator \( T_P \).

The following was reported in [51].

\(^1\)This notion is due to [15]. We stick to the old terminology, although quasi-interpretations should really be thought of as, and indeed are, programs with negative body literals only.
Theorem 1. For any normal program $P$ and (two-valued) interpretation $I$ we have

$$GL_P(I) = T_{\text{fin}}(P)(I)$$

Proof. We show first that for every $A \in GL_P(I)$ there exists a clause in $\text{fix}(P)$ with head $A$ whose body is true in $I$, which implies $A \in T_{\text{fin}}(P)(I)$. We show this by induction on the powers of $T_{P/I}$; recall that $GL_P(I) = T_{P/I}^{\omega}$.

For the base case $T_{P/I}^{\omega} = \emptyset$ there is nothing to show.

So assume now that for all $A \in T_{P/I}^{\omega}$ there exists a clause in $\text{fix}(P)$ with head $A$, whose body is true in $I$. For $A \in T_{P/I}^{\omega}(n)$ there exists a clause $A_1 \ldots A_n$ in $P/I$ such that $A_1 \ldots A_n \notin T_{P/I}^{\omega}$, hence by construction of $P/I$ there is a clause $A_1 \ldots A_n \neg B_1 \ldots 
eg B_m$ in ground($P$) with $B_1 \ldots B_m \notin I$. By induction hypothesis we obtain that for each $i = 1 \ldots n$ there exists a clause $A_i$ whose body is true in $I$, hence $A_i \in T_{\text{fin}}(P)(I)$. So by definition of $T_P$ the clause $A_1 \ldots A_n \neg B_1 \ldots 
eg B_m$ is contained in $\text{fix}(P)$. From $I \models \text{body}_i$ and $B_1 \ldots B_m \notin I$ we obtain $A \in T_{\text{fin}}(P)(I)$, as desired.

This closes the induction argument and we obtain $GL_P(I) = T_{\text{fin}}(P)(I)$.

Now conversely, assume that $A \in T_{\text{fin}}(P)(I)$. We show that $A \in GL_P(I)$ by proving inductively on $k$ that $T_{T_{P/I}^{\omega}}^k(I) \models GL_P(I)$ for all $k \in \mathbb{N}$.

For the base case, we have $T_{T_{P/I}^{\omega}}^0(I) = \emptyset$ so there is nothing to show.

So assume now that $T_{T_{P/I}^{\omega}}^k(I) \models GL_P(I)$, and let $A \in T_{T_{P/I}^{\omega}(k+1)}(I) \setminus T_{T_{P/I}^{\omega}}^k(I)$. Then there exists a clause $A_1 \ldots A_n \neg B_1 \ldots 
eg B_m$ in $T_{P/I}^{\omega}(k)$ whose body is true in $I$. Thus $B_1 \ldots B_m \notin I$ and for each $i = 1 \ldots n$ there exists a clause $A_i$ with body in $T_{P/I}^{\omega}$ true in $I$. So $A_i \in T_{T_{P/I}^{\omega}}^k(I)$, as desired. Furthermore, by definition of $T_P$ there exists a clause $A_1 \ldots A_n \neg B_1 \ldots 
eg B_m$ in ground($P$), and since $B_1 \ldots B_m \notin I$ we obtain $A \in GL_P(I)$, and hence $T_{T_{P/I}^{\omega}(k+1)}(I) \models GL_P(I)$. This closes the induction argument and we obtain $T_{\text{fin}}(P)(I) \models GL_P(I)$. □

The proof of Theorem 1 is taken directly from [52], which appeared in compressed form as [51]. We have included it here for completeness of the exhibition and because the result is central for the rest of this paper. This correspondence can also be carried over to the Fitting/well-founded semantics. More precisely, the following was shown in [51], from which Theorem 1 is an easy Corollary.

Theorem 2. For any normal program $P$ and any three-valued interpretation $I$ we have $\Psi_P(I) = \Phi_{\text{fin}}(P)(I)$ where $\Psi_P$ is the operator due to [6] used for characterizing three-valued stable models of $P$ and $\Phi_{\text{fin}}(P)$ is the operator from [16] used for characterizing the Fitting or Kripke-Kleene semantics of $\text{fix}(P)$.

We do not include details on this result here since we will need it only in passing in the sequel. The interested reader should consult [51]. A corollary from the result just mentioned is that the well-founded model of some given program $P$ coincides with the Fitting model of $\text{fix}(P)$.
3 Continuity

Theorem 1 enables us to carry over results on the single-step operator, respectively on the supported-model semantics, to the Gelfond-Lifschitz operator respectively the stable-model semantics. The following observation is of technical importance.

**Proposition 1.** Let $P$ be a definite program, $A \in B_P$ and $n \in \mathbb{N}$. Then $A \in T_P \uparrow \! n$ if and only if $A$ is a clause in $T_P \uparrow \! n$.

**Proof.** Let $A \in T_P \uparrow \! n$ for some $n \in \mathbb{N}$. We proceed by induction on $n$. If $n = 1$, then there is nothing to show. So assume that $n > 1$. Then there is a clause $A \in \text{body in ground}(P)$ such that all atoms $B_i$ in body are contained in $T_P \uparrow \! (n - 1)$, and by induction hypothesis there are clauses $B_i$ in $T_P \uparrow \! (n - 1)$. Unfolding these clauses with $A$ body shows that $A$ is also contained in $T_P \uparrow \! n$.

Conversely, assume there is a clause $A \in T_P \uparrow \! n$. We proceed again by induction. If $n = 1$, there is nothing to show. So let $n > 1$. Then there exists a clause $A = A_1 \ldots A_k$ in $\text{ground}(P)$ and clauses $A_i \in T_P \uparrow \! (n - 1)$. By induction hypothesis, we obtain $A_i \in T_P \uparrow \! (n - 1)$ for all $i$, and hence $A \in T_P \uparrow \! n$. \[ \square \]

Since the single-step operator is not monotonic in general, several authors have made use of metric-based [17,18,22,25–27,29,46] or even topological [3,4,22,24,43,45,47] methods for obtaining fixed-points and hence supported models of the programs in question. Central to these investigations is the Cantor topology $Q$ on $I_P$, which was studied as the query topology in [4] and in more general terms as the atomic topology in [45]. It is the product topology on $\{t,f\}^{B_P}$, where the set of truth values $\{t,f\}$ is endowed with the discrete topology, and we refer to [33] for basic notions of topology. A subbase of the Cantor topology can be given as

$$\{\{I \in I_P \mid I \models L\} \mid L \text{ is a ground literal}\}$$

which was noted in [45]. We can now employ Theorem 1 to carry over some of these results to the treatment of the Gelfond-Lifschitz operator and the stable semantics.

Given a program $P$, we know by Theorem 1 that $\text{GL}_P$ is continuous at some $I \in I_P$ in $Q$ if and only if $T_{\text{fix}(P)}$ is continuous at $I$. This gives rise to the following theorem.

**Theorem 3.** Let $P$ be a normal logic program and let $I \in I_P$. Then $\text{GL}_P$ is continuous at $I$ in $Q$ if and only if whenever $\text{GL}_P(I)(A) = f$ then either there is no clause with head $A$ in $\text{ground}(P)$ or there exists a finite set $S(I, A) = \{A_1, \ldots, A_k\} \subseteq B_P$ such that $I(A_i) = t$ for all $i$ and for every clause $A \in \text{body in ground}(P)$ at least one $\lnot A_i$ or some $B$ with $\text{GL}_P(I)(B) = f$ occurs in body.
Proof. The proof is based on the characterization of continuity of the $T_P$-operator given in [45], in the formulation which can be found in [29, Theorem 45], which reads as follows.

The single-step operator $T_P$ is continuous in $Q$ if and only if, for each $I \in I_P$ and for each $A \in B_P$ with $A \in T_P(I)$, either there is no clause in $P$ with head $A$ or there is a finite set $S(I,A) = \{A_1, \ldots, A_n\}$ of elements of $B_P$ with the following properties:

(i) $A_1, \ldots, A_n \in I$ and $B_1, \ldots, B_n \in I$.
(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 $C$.

Using this and Theorem 1, and by observing that there are no positive body atoms occurring in $\text{fix}(P)$, we obtain the following:

$GL_P$ is continuous at $I$ if and only if whenever $GL_P(I)(A) = t$, then either there exists no clause with head $A$ in $\text{fix}(P)$ or there exists a finite set $S(I,A) = \{A_1, \ldots, A_n\} \subseteq B_P$ such that $I(A_i) = t$ for all $i$ and for every clause $A$ body in $\text{fix}(P)$ at least one $\neg A_i$ occurs in body. Now let $A \quad B_1, \ldots, B_n \quad \neg C_1, \ldots, \neg C_m$ be a clause in $\text{ground}(P)$ and assume that no $\neg A_i$ occurs in its body. We show that there is some $B_i$ with $GL_P(I)(B_i) = t$.

We have $B_i \in GL_P(I) = T_P|I \uparrow \omega$. As in the proof of Proposition 1 we derive that there is a clause $A \quad \neg D_1, \ldots, \neg D_n \quad \neg C_1, \ldots, \neg C_m$ in $\text{fix}(P)$ with $D_j \notin I$ for all $j = 1, \ldots, n$. Since the clause $A \quad \neg D_1, \ldots, \neg D_n \quad \neg C_1, \ldots, \neg C_m$ is contained in $\text{fix}(P)$, we know that some atom from the set $S(I,A)$ must occur in its body. It cannot occur as any $D_j$ because $I(D_j) = t$ for all $j$. It also cannot occur as any $C_i$ by assumption. So we obtain a contradiction, which finishes the argument.

Conversely, let $P$ be such that the condition on $GL_P$ in the statement of the theorem holds. We will again make use of the observation made at the beginning of this proof. So let $A \in B_P$ with $GL_P(I)(A) = t$. If there is no clause with head $A$ in $\text{fix}(P)$, then there is nothing to show. So assume that there is a clause with head $A$ in $\text{fix}(P)$. Then there is a clause with head $A$ in $P$, and by assumption we know that there exists a finite set $S(I,A) = \{A_1, \ldots, A_n\} \subseteq B_P$ such that $I(A_i) = t$ for all $i$ and for every clause $A$ body in $\text{ground}(P)$ at least one $\neg A_i$ or some $B$ with $GL_P(I)(B) = t$ occurs in body. Now let $A \quad \neg B_1, \ldots, \neg B_m$ be a clause in $\text{fix}(P) = T_P|I \uparrow \omega$, i.e. there is $k \in \mathbb{N}$ with $A \quad \neg B_1, \ldots, \neg B_m$ contained in $T_P|I \uparrow k$. Note that $\pi = 0$ is impossible since this would imply $GL_P(I)(A) = t$ contradicting the assumption on $A$. We proceed by...
induction on k. If \( k = 1 \), then \( A \neg B_1 \neg B_n \) is contained in \( \text{ground}(P) \), hence one of the \( B_j \) is contained in \( S(I, A) \) which suffices. For \( k > 1 \), there is a clause \( A C_1 \ldots C_m \neg D_1 \ldots \neg D_m \) in \( \text{ground}(P) \) and clauses \( C_i \) body in \( T^P_j(k \rightarrow 1) \) which unfold to \( A \neg B_1 \neg B_n \). By assumption we either have \( D_j \in S(I, A) \) for some \( j \), in which case there remains nothing to show, or we have that \( \text{GL}_P(I)(C_i) = \text{f} \) for some \( i \). In the latter case we obtain that body is non-empty by an argument similar to that of the proof of Proposition 1, so by assumption there is a (negated) atom in body, and hence in \( \{ B_1 \ldots n \} \), which is also in \( S(I, A) \), which finishes the proof. \( \Box \)

We can also observe the following special instance. A local variable is a variable occurring in some clause body but not in the corresponding head.

**Corollary 1.** Let \( P \) be a normal program without local variables. Then \( \text{GL}_P \) is continuous in \( Q \).

**Proof.** We employ Theorem 3. Let \( I \in I_P \) and \( A \in B_P \) with \( \text{GL}_P(I)(A) = \text{f} \). Since \( P \) has no local variables, it is of finite type. So the set \( B \) of all negated body atoms in clauses with head \( A \) is finite. Let \( S(I, A) = \{ B \in B \mid I(B) = \text{f} \} \), which is also finite. If each clause with head \( A \) contains some negated atom from \( S(I, A) \), then there is nothing to show. So assume there is a clause \( A C_1 \ldots C_m \neg D_1 \ldots \neg D_m \) in \( \text{ground}(P) \) with \( B_j \in S(I, A) \) for all \( j \), i.e. \( I(B_j) = \text{t} \) for all \( j \). But then \( A A_1 \ldots A_n \) is a clause in \( P/I \) and \( A \in T^P/I \uparrow \omega \), which implies that there is some \( i \) with \( A_i \in T^P/I \uparrow \omega = \text{GL}_P(I) \), which finishes the argument by Theorem 3. \( \Box \)

Measurability is much simpler to deal with.

**Theorem 4.** Let \( P \) be a normal program. Then \( \text{GL}_P \) is measurable with respect to the \( \sigma \)-algebra \( \sigma(Q) \) generated by \( Q \).

**Proof.** By [28, Theorem 2], which states that \( T_P \) is measurable with respect to \( \sigma(Q) \) for all \( P \), we obtain that \( T_{\text{lex}}(P) \) is measurable with respect to \( \sigma(Q) \), and by Theorem 1 we know that \( T_{\text{lex}}(P) \equiv \text{GL}_P \). \( \Box \)

## 4 Obtaining models

As already mentioned above, topological methods in logic programming can for example be used for obtaining models of programs iteratively, although the underlying operator is not monotonic. The following variant of [29, Theorem 44] can be proven directly.

**Theorem 5.** Let \( P \) be a normal program and let \( \text{GL}_P \) be continuous and such that the sequence of iterates \( \text{GL}_P^n(I) \) converges in \( Q \) to some \( M \in I_P \). Then \( M \) is a stable model of \( P \).
Proof. By continuity we obtain

\[ M = \lim \text{GL}^n_P(I) = \text{GL}_P(\lim \text{GL}^n_P(I)) = \text{GL}_P(M) \]

\( \square \)

We can also employ knowledge about relationships between the single-step operator and the Fitting operator [16]. The latter is defined on three-valued interpretations, which consist of sets of ground literals (instead of ground atoms) which do not contain complementary literals. As such, they carry set-inclusion as an ordering, which renders the space \( I_{P,3} \) of all three-valued interpretations a complete partial order (cpo). It is in fact exactly the Plotkin domain \( T^* \) due to [41]. Alternatively, we can understand three-valued interpretations as mappings from atoms to the set \( \{ \text{false}, \text{true}, \text{undefined} \} \) of truth values, where \( \text{undefined} \) stands for undefined or undetermined. The Fitting operator \( \Phi_P \), for given program \( P \), is now defined as a function \( \Phi_P : I_{P,3} \to I_{P,3} \) such that for all \( A \in B_P \), for which there exists a clause \( A \models L_i \) in \( \text{ground}(P) \) with \( L_i \models \text{ground}(P) \) and \( \text{ground}(P) \) contains all \( \neg A \) such that for all clauses \( A \models L_i \), there is at least one \( L_i \models I \). It was shown in [16] that \( \Phi_P \) is a monotonic operator on \( I_{P,3} \).

If \( I \) is a three-valued interpretation, then \( I^+ \) denotes the two-valued interpretation assigning truth value \( t \) to exactly those atoms which are true in \( I \).

**Proposition 2.** Let \( P \) be a normal program and assume that the well-founded model \( M \) of \( P \) is total (i.e. every atom is true or false in it). Then \( \text{GL}^n_P(\emptyset) \) converges in \( Q \) to \( M^+ \) and \( M^+ \) is the unique stable model of \( P \).

**Proof.** This follows immediately from Theorem 1 and [24, Theorem 4.4], which shows the following.

If \( M = \Phi_R \uparrow \omega \) is total, then \( T^*_R(\emptyset) \) converges in \( Q \) to \( M^+ \), and \( M^+ \) is the unique supported model of \( R \).

\( \square \)

Metric-based approaches also carry over. A level mapping is a mapping from \( B_P \) to some ordinal \( \alpha \). A program \( P \) is locally stratified [44] if there exists a level mapping \( l : B_P \to \alpha \), where \( \alpha \) is some ordinal, such that for each clause \( A \models B_i \) in \( \text{ground}(P) \) and \( A \models B_j \) for all \( i \) and \( j \). It is called locally hierarchical [10], if additionally \( l(A) > l(A_i) \) for all \( i \). Given a level mapping \( l : B_P \to \alpha \), we denote by \( I_l \) the set of all symbols \( 2^{-\beta} \) for \( \beta \leq \alpha \), ordered by \( 2^{-\beta} < 2^{-\gamma} \) if \( \gamma < \beta \). \( I_l \) can be understood as a subset of the reals if \( \alpha = \omega \), i.e. if \( l \) maps into the natural numbers. For two (two-valued) interpretations \( I \) and \( J \), we define \( d_l(I, J) = 2^{-\beta} \), where \( \beta \) is the least ordinal such that there is an atom of level \( \beta \) on which \( I \) and \( J \) disagree. If \( \alpha = \omega \), then \( d_l \) is an ultrametric on \( B_P \), and this construction was put to use e.g. in [17]. In the general case, \( d_l \) is a generalized ultrametric on \( B_P \), as used in logic programming e.g. in [25, 29, 43]. A mapping \( f \) is called strictly contracting with respect to a generalized ultrametric \( d \) if \( d(f(x), y) < d(x, y) \) for all \( x, y \) with \( f
\(x = y\). Strictly contracting mappings have unique fixed points if the underlying generalized ultrametric space satisfies a completeness condition called spherical completeness [43].

**Theorem 6.** Let \(P\) be locally stratified with corresponding level mapping \(l\). Then \(\text{GL}_P\) is strictly contracting with respect to \(d_l\) which is spherically complete. If \(l\) maps to \(\omega\) then \(\text{GL}_P\) is a contraction with respect to \(d_l\). Furthermore, in both cases, \(\text{GL}_P\) has a unique fixed point and \(P\) has a unique stable model.

**Proof.** If \(P\) is locally stratified with respect to \(l\), then \(\text{fix}(P)\) is locally hierarchical with respect to \(l\). It thus suffices to apply Theorem 1 in conjunction with Theorem [47, Theorem 3.8], which shows the following.

\[
\text{Let } R \text{ be a normal logic program which is locally hierarchical with respect to a level mapping } l : B_R \rightarrow \gamma. \text{ Then } T_R \text{ is strictly contracting with respect to the generalized ultrametric } d_l \text{ induced by } l. \text{ Therefore, } T_R \text{ has a unique fixed point and hence } R \text{ has a unique supported model.}
\]

\(\square\)

With the remarks already made on the fact that the well-founded model of some given program \(P\) coincides with the Fitting model of \(\text{fix}(P)\), for any normal program \(P\), we can also derive the following result. **Dislocated generalized ultrametric spaces** are defined by relaxing one of the defining conditions on generalized ultrametrics, for details see [29]. Strictly contracting mappings can be defined analogously, and have similar properties.

**Theorem 7.** Let \(P\) be a program with total well-founded model \(I \cup \neg(B_P \setminus I)\) with \(I \cap B_P\). Then \(\text{GL}_P\) is strictly contracting on the spherically complete dislocated generalized ultrametric space \((I_P, d_{I_P})\) where we have \(d_{I_P}(J, K) = \max\{d_J(J, K)\} \downarrow(I, K)\) for all \(J \in I_P\) (\(d\) is defined by \(l(A)\)). Let \(R\) be the minimal \(\alpha\) such that \(\Phi_{\text{fix}(P)}\uparrow(\alpha)(A) = I(A)\) for some \(A\).

**Proof.** The program \(P\) has a total well-founded model, which implies that \(\text{fix}(P)\) has a total Fitting model. So \(l\) as given by the statement is well-defined, and \(\text{fix}(P)\) is \(\Phi\)-accessible in the sense of [29]. Now apply [29, Proposition 41], which shows that \(T_P\) is strictly contracting for every \(\Phi\)-accessible program. \(\square\)

### 5 Neural-symbolic integration

Intelligent systems based on logic programming on the one hand, and on artificial neural networks (sometimes called connectionist systems) on the other, differ substantially. Logic programs are highly recursive and well understood from the perspective of knowledge representation: The underlying language is that of first-order logic, which is symbolic in nature and makes it easy to encode problem specifications directly as programs. The success of artificial neural networks lies in the fact that they can be trained using raw data, and in some problem domains
the generalization from the raw data made during the learning process turns out
to be highly adequate for the problem at hand, even if the training data contains
some noise. Successful architectures, however, often do not use recursive (or
recurrent) structures. Furthermore, the knowledge encoded by a trained neural
network is only very implicitly represented, and no satisfactory methods for
extracting this knowledge in symbolic form are currently known.

It would be very desirable to combine the robust neural networking machinery
with symbolic knowledge representation and reasoning paradigms like logic
programming in such a way that the strengths of either paradigm will be re-
tained. Current state-of-the-art research, however, fails by far to achieve this
ultimate goal. As one of the main obstacles to be overcome we perceive the
question how symbolic knowledge can be encoded by artificial neural networks:
Satisfactory answers to this will naturally lead the way to knowledge extraction
algorithms and to hybrid neural-symbolic systems.

Earlier attempts to integrate logic and connectionist systems have mainly
been restricted to propositional logic, or to first-order logic without function
symbols. They go back to the pioneering work by McCulloch and Pitts [39], and
have led to a number of systems developed in the 80s and 90s, including Towell
and Shavlik's KBANN [49], Shastri's SHRUTI [48], the work by Pinkas [40],
Hölldobler [30], and d'Avila Garcez et al. [12,14], to mention a few, and we refer
to [8,13,21] for comprehensive literature overviews.

Without the restriction to the finite case (including propositional logic and
first-order logic without function symbols), the task becomes much harder due
to the fact that the underlying language is infinite but shall be encoded using
networks with a finite number of nodes. The sole approach known to us for
overcoming this problem (apart from work on recursive autoassociative memory,
RAAM, initiated by Pollack [42], which concerns the learning of recursive terms
over a first-order language) is based on a proposal by Hölldobler et al. [32],
spelled out first for the propositional case in [31], and reported also in [23]. It
is based on the idea that logic programs can be represented — at least up to
subsumption equivalence [38] — by their associated single-step operators. Such
an operator can then be mapped to a function on the real numbers, which can
under certain conditions in turn be encoded or approximated e.g. by feedforward
networks with sigmoidal activation functions using an approximation theorem
due to Funahashi [19].

We will carry over this result to the Gelfond-Lifschitz operator and the stable
model semantics. Since the topology $Q$ introduced earlier is homeomorphic to the
Cantor topology on the real line [45], there exists a homeomorphism $\iota : I_P \to C$,
where $C$ is the Cantor set within the unit interval, endowed with the subspace
topology inherited from the reals. We can thus embed any function $f : I_P \to I_P$
which is continuous in $Q$ as a continuous function $\iota(f) : C \to C : \iota(f)(x) =
\iota(f(\iota^{-1}(x)))$. By well-known results, e.g. [19] as mentioned earlier, such functions
can be approximated uniformly by artificial neural networks in many different
network architectures.
Theorem 8. Let $P$ be a normal logic program. Then $GL_P$ can be approximated almost everywhere up to an arbitrarily chosen error bound by input-output functions of three-layer feedforward neural networks with sigmoidal activation functions. If $GL_P$ is furthermore continuous in $Q$ then uniform approximation is possible on all of $C$.

Proof. We use Theorem 1. The first statement then follows from Theorem 4 together with a result from [33] saying that each measurable function can be approximated almost everywhere by three-layer feedforward networks in the indicated way — see also [28, Theorem 7]. The second statement follows from [19] or from [28, Theorem 5].

The references mentioned in the proof of Theorem 8 provide further results, in particular on error bounds, and they can also be carried over straightforwardly.

Another improvement on the basic results by Hölldobler et al [32] employed an alternative network architecture. In [2], results were provided for encoding and approximating $\mathcal{I}(T_P)$ by iterated function systems, which in turn could be encoded using a recurrent neural networks structure. The advantage of this approach is that algorithms for constructing approximating networks can be given explicitly, in contrast to the results in [23, 28, 32]. These results also hinge on continuity or Lipschitz-continuity of $\mathcal{I}(T_P)$ with respect to the Cantor topology only, and can be carried over to the Gelfond-Lifschitz operator in a straightforward way. The paper [5] provides related results using cellular automata, treating logic programs without local variables — a property which also carries over to the fixpoint completion. Hence these results carry over mutatis mutandis to the Gelfond-Lifschitz operator.

6 Conclusions

We have displayed the usefulness of the results reported in [51] to the operator-based analysis of knowledge representation under the stable semantics. We have shown that many results from the study of the supported-model semantics by means of the single-step operator can be carried over to the stable semantics almost without effort.

Our results are of a theoretical nature, and we do not propose to study them for implementation purposes. The idea to use the fixpoint completion for obtaining stable models (or similar constructions for obtaining answer sets or well-founded models etc.) of programs is already folklore knowledge in the community, and need not be further mentioned. The emphasis of our exhibition is on the observation that not only models, but also corresponding semantic operators are related by means of the fixpoint completion, and on the aspects which this new insight allows to study.

Our observations are valid for first-order languages including function symbols, a syntax whose study is often neglected in the non-monotonic reasoning community. It is not at all surprising, that for finite languages alternative methods of program transformation can be found, which allow for efficient computation of stable models [34–36].
References


^2^ Available online as [52].