

Acceptable Programs Revisited

Pascal Hitzler¹ and Anthony Karel Seda

*Department of Mathematics
National University of Ireland, Cork
Cork, Ireland*

Abstract

Acceptable logic programs have been studied extensively in the context of proving termination of Prolog programs. It is difficult, however, to establish acceptability from the definition since this depends on finding a suitable model, which need not be a Herbrand model in general, together with a suitable level mapping that one can use to check the conditions which characterize acceptability. In this paper, we will see that when working over a fixed but arbitrary preinterpretation, a method can be provided for obtaining both a suitable model and a canonical level mapping which are sufficient for this purpose. Furthermore, the canonical model and level mapping obtained will turn out to be sufficient for discussing termination of non-ground queries.

1 Introduction

Acceptable programs were first studied in detail in [3] where they were shown to coincide, if floundering is ignored, with the programs which are left-terminating. They have been much examined in the literature, for example in [1,4,9,11,15], and they form an important class of programs. However, in order to show from the definition that a given program P is acceptable, it is necessary to determine a level mapping and a model I for P which satisfy the conditions of the definition, see Definition 3.2. But this may be difficult to do, especially as the model I given in the definition need not be a Herbrand model in general. The notion of semi-acceptability as given in [4] somewhat simplifies finding a suitable level mapping but is not much more natural in some cases (cf. the discussion in [9]) and a suitable model still needs to be found. It is therefore desirable to simplify these tasks, if possible, and it is the purpose of this paper to take some steps in this direction.

¹ The first named author acknowledges financial support under grant SC/98/621 from Enterprise Ireland.

After establishing preliminaries in Section 2, we will start in Section 3 by displaying some of the difficulties which underlie the general notion of acceptability. In particular, we will show that central notions and results on acceptable programs depend heavily on the choice of preinterpretation (including the choice of an underlying first order language). We will also briefly consider the usefulness of constructive negation in the sense of [8] as an alternative way of handling non-ground negative literals, an approach taken in [15]. The problems pointed out in Section 3 will guide the rest of the paper.

In Section 4, we will generalize a result from [3] and prove that for any fixed preinterpretation a unique supported model can be found for any given program which is acceptable in that preinterpretation. (Precisely what is meant here will be defined in Section 3). Indeed, it turns out that this model can be determined as the limit of the iterates of the single-step operator, where this limit is taken in the atomic topology Q as introduced in [17], see Definition 2.1 herein. The importance of this observation lies in the fact that convergence in the atomic topology Q can be characterized in a very simple way, see again Definition 2.1.

In Section 5, we will introduce the notion and construction of a canonical level mapping for a given acceptable program.

The main result of this paper, Theorem 6.3, is a characterization of acceptable programs by means of the unique supported model and the canonical level mapping just mentioned. The importance of this characterization is that both the model and the level mapping can be obtained by a constructive, if not effective, approach². In the same section, Section 6, we will also obtain strong minimality results for the unique supported model and the canonical level mapping for a given acceptable program.

Finally, in Section 7, we will show that the canonical level mapping and the unique supported model are suitable and sufficient for a fully general discussion of the termination of non-ground queries as in [3].

Acceptable programs have unique supported models – a result which was provided in [3] for the case of Hebrand preinterpretations and which will be generalized to arbitrary preinterpretations in the sequel. We call classes of programs with this property *unique supported model classes*, and of course they leave little doubt about their intended semantics. The present paper is a side-product of work in progress by the authors which aims at obtaining a deeper understanding of these classes by comparing different methods for obtaining them, see [13].

2 Preliminaries

Our notation basically follows [14], and we will give next a short review of the main terminology used.

² This, however, still leaves the problem of finding a suitable preinterpretation unsolved.

A *preinterpretation* J for a normal logic program P consists of a first order language \mathcal{L} which contains the constant, function and predicate symbols occurring in P (but not the equality symbol) together with a domain \mathcal{D} and assignments, as usual, of constant and function symbols in \mathcal{L} to constants and functions respectively in \mathcal{D} . An interpretation (model) I for P based on J will be called a *J -interpretation* (*J -model*) for P and, additionally, assigns predicate symbols in \mathcal{L} to relations in \mathcal{D} in the usual way. Variable assignments which map into the domain of J will be called *J -variable assignments*. It should be noted at this point that our definition of *preinterpretation*, and therefore of *interpretation*, departs from normal practice in that we include \mathcal{L} in the definition. The reason for doing this will become clear when we define J -acceptability later, and it is mainly a convenient device for fixing \mathcal{L} since its choice affects so much whether or not a program is acceptable.

Following [14] and §2 of [17], we denote by $B_{P,J}$ the set of all J -instances of predicate symbols in \mathcal{L} i.e. the set of all $p(d_1, \dots, d_n)$, where p is an n -ary predicate symbol in \mathcal{L} and $d_1, \dots, d_n \in \mathcal{D}$. An element $A = p(d_1, \dots, d_n)$ of $B_{P,J}$ is called a *J, v -(ground) instance* or *J -(ground) instance* of an atomic formula $A' = p(t_1, \dots, t_n)$ in \mathcal{L} if there exists a J -variable assignment v such that $A' \mid v = A$, meaning that $t_i \mid v = d_i$ for $i = 1, \dots, n$, where $t \mid v$ is the denotation of a term t relative to J and v . Since each $t_i \mid v \in \mathcal{D}$, any J -instance of A' is variable free. This extends easily to literals L , where $L = \neg A' = \neg p(t_1, \dots, t_n)$, say. Thus, the symbol $\neg p(d_1, \dots, d_n)$ is called a *J, v -(ground) instance* or *J -(ground) instance* of the literal L if there exists a J -variable assignment v such that $p(t_1, \dots, t_n) \mid v = p(d_1, \dots, d_n)$. We sometimes loosely refer to J -ground instances of atoms and of literals as *J -ground atoms* and *J -ground literals* respectively. In accordance with Definition 1 of [17], we write $\text{ground}_J(P)$ for the set of all J -(ground) instances of clauses, or J -ground clauses, in P . Thus, typically, if $A' \leftarrow L_1, \dots, L_n$ is a clause in P , then $A' \mid v \leftarrow L_1 \mid v, \dots, L_n \mid v$ is an element of $\text{ground}_J(P)$, where v is a J -variable assignment such that $A = A' \mid v$ is a J -instance of A' and $L_i \mid v$ is a J -instance of L_i for $i = 1, \dots, n$. All elements of $\text{ground}_J(P)$ are obtained thus for some clause and some J -variable assignment.

The set of all J -interpretations for a given normal program P will be denoted by $I_{P,J}$, and this set is a complete lattice with respect to the ordering \subseteq defined by $I \subseteq J$ if and only if $I \models A$ implies $J \models A$ for every $A \in B_{P,J}$. As usual, we identify $I_{P,J}$ with the power set $2^{B_{P,J}}$ and the ordering \subseteq is then indeed set-inclusion. For $I \in I_{P,J}$, we set ${}^c I = B_{P,J} \setminus I$. With this convention and following §2 of [17], in classical two-valued logic we write $I \models p(d_1, \dots, d_n)$ (resp. $I \models \neg p(d_1, \dots, d_n)$) if $p(d_1, \dots, d_n) \in I$ (resp. $p(d_1, \dots, d_n) \notin I$). By abusing conjunction in the obvious way (see §2 of [17]), it is now meaningful to write $I \models L_1 \mid v, \dots, L_n \mid v$, where $L_1 \mid v, \dots, L_n \mid v$ denotes a “conjunction” of J -instances of literals.

The immediate consequence operator $T_{P,J}$ for a given program P is defined

Table 1
Truth table for Kleene’s strong three-valued logic.

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

as a mapping on $I_{P,J}$: for $I \in I_{P,J}$ define $T_{P,J}(I)$ as the set

$$\{A \in B_{P,J} \mid \text{there exists } A \leftarrow \mathbf{body} \text{ in } \text{ground}_J(P) \text{ with } I \models \mathbf{body}\},$$

where \mathbf{body} denotes a conjunction of J -instances of literals. Recall from [2] that a J -interpretation M is a supported model of P if and only if $T_{P,J}(M) = M$ if and only if³ M is a model of the Clark-completion $\text{comp}(P)$ of P .

Following [10], a *partial J -interpretation* I is a pair (I^+, I^-) of subsets of $B_{P,J}$ such that I^+ and I^- are disjoint. Partial interpretations are interpreted in Kleene’s strong three-valued logic⁴ following [10]. A partial interpretation (I^+, I^-) is called *total* if $I^+ \cup I^- = B_{P,J}$, and such an interpretation can be naturally identified with an element of $I_{P,J}$. The set $I_{P,J,3}$ of all partial J -interpretations is a complete partial order, indeed complete semi-lattice, under the ordering: $(I_1^+, I_1^-) \leq (I_2^+, I_2^-)$ iff $I_1^+ \subseteq I_2^+$ and $I_1^- \subseteq I_2^-$, where we take the bottom element to be $\perp = (\emptyset, \emptyset)$. Total interpretations are in fact maximal elements in the given ordering.

The three-valued operator $\Phi_{P,J}$ is defined as a mapping on partial J -interpretations K as follows. We set $\Phi_{P,J}(K) = (I^+, I^-)$, where I^+ is the set of all $A \in B_{P,J}$ with the property that there exists a clause $A \leftarrow \mathbf{body}$ in $\text{ground}_J(P)$ such that \mathbf{body} is true in K , and I^- is the set of all $A \in B_{P,J}$ such that for all clauses $A \leftarrow \mathbf{body}$ in $\text{ground}_J(P)$ we have that \mathbf{body} is false

³ Strictly speaking, this is not correct, since a model of the Clark-completion must also satisfy Clark’s equality theory, see [14]. The given identification is common practice, though.

⁴ Given a partial interpretation $I = (I^+, I^-)$, atoms in I^+ carry the truth value *true* (t) in I and atoms in I^- the value *false* (f) in I . Atoms which are neither in I^+ nor in I^- carry the truth value *undefined* (u). For our purposes, we only need to know how conjunction and negation operate in this three-valued logic, and this is given by the truth table in Table 1.

in K . We note that $\Phi_{P,J}$ is monotonic, and hence we define

$$\begin{aligned}\Phi_{P,J} \uparrow 0 &= \perp, \\ \Phi_{P,J} \uparrow (k+1) &= \Phi_{P,J}(\Phi_{P,J} \uparrow k) \text{ for any ordinal } k, \text{ and} \\ \Phi_{P,J} \uparrow \alpha &= \sup\{\Phi_{P,J} \uparrow \beta \mid \beta < \alpha\} \text{ for a limit ordinal } \alpha.\end{aligned}$$

If J is a Herbrand-preinterpretation, we will omit the subscript J and write $\text{ground}(P)$, T_P , Φ_P etc. instead of $\text{ground}_J(P)$, $T_{P,J}$, $\Phi_{P,J}$ etc.

Among other things, we will be concerned with convergence (in the topological sense) of sequences of iterates of the immediate consequence operator, and we will therefore need the following definition and results of [17].

Definition 2.1 Fixing a preinterpretation J , the *atomic topology* Q is defined on the space $I_{P,J}$ of all interpretations based on J and is characterized via convergence as follows. A net⁵ (I_λ) of interpretations converges in Q if and only if, for every J -ground atom A , there exists an index α such that either $I_\beta \models A$ for all $\beta > \alpha$ or for all $\beta > \alpha$ we have $I_\beta \not\models A$, i.e. if every J -ground atom A is either eventually true in I_λ or is eventually false in I_λ . The unique limit of a converging net is the interpretation which assigns the truth value true to all J -ground atoms which are eventually true in I_λ .

The atomic topology was introduced in [17] and is a generalization of the query topology discussed in [5] and defined on the space of all Herbrand-interpretations for a given program. The importance of the atomic topology for logic programming semantics is obvious from the following theorem which is a generalization of [12, Theorem 3.6].

Theorem 2.2 *Let P be a normal logic program, let J be a preinterpretation and let I be a J -interpretation for P . If the sequence of iterates $(T_{P,J}^n(I))_{n \in \mathbb{N}}$ of the immediate consequence operator $T_{P,J}$ converges in Q to some M , then M is a J -model for P . If the sequence $(T_{P,J}^n(I))$ does not converge in Q for any J -interpretation I , then P has no supported J -models.*

Proof. Suppose $T_{P,J}^n(I) \rightarrow M$ in Q for some J -interpretation I . We have to show that $T_{P,J}(M) \subseteq M$. Let $A \in T_{P,J}(M)$. By definition of $T_{P,J}$, there exists a J -ground clause $A \leftarrow A_1, \dots, A_{k_1}, \neg B_1, \dots, \neg B_{l_1}$ where, for $k = 1, \dots, k_1$ and for $l = 1, \dots, l_1$, we have $M \models A_k$ and $M \not\models B_l$. By convergence in Q , there is an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and for $k = 1, \dots, k_1, l = 1, \dots, l_1$, we have $A_k \in T_{P,J}^n(I)$ and $B_l \notin T_{P,J}^n(I)$. By definition of $T_{P,J}$, it follows that we have $A \in T_{P,J}^m(I)$ for all $m \geq n_0 + 1$. Hence, $A \in T_{P,J}^n(I)$ eventually and therefore, by convergence in Q again, $M \models A$, which proves the first statement.

Now, if M is a supported J -model for P , then $(T_{P,J}^n(M))$ is constant with value M , and so the second statement is trivially true. \square

⁵ The topological notion of a net (see [20] or any book on general topology) is necessary in order to characterize Q if the domain \mathcal{D} of the preinterpretation is uncountable. For our purpose, it is sufficient to think of nets as sequences.

3 Motivation

It will be convenient to first give the formal definition of acceptability. Given a normal logic program P and a preinterpretation J , a *level mapping* for P is a mapping l from $B_{P,J}$ to the set of natural numbers. We will always assume that l is in fact extended to J -instances of literals by setting $l(\neg A) = l(A)$ for every $A \in B_{P,J}$.

Following [3], we define a subset P^- of P as follows.

Definition 3.1 Let P be a normal logic program and let p, q be predicate symbols occurring in P .

- (i) We say that p *refers to* q if there is a clause in P with p in its head and q in its body.
- (ii) We say that p *depends on* q if (p, q) is in the reflexive, transitive closure of the relation *refers to*.
- (iii) The set of predicate symbols in P which occur in a negative literal in the body of a clause in P is denoted by Neg_P .
- (iv) The set of predicate symbols in P on which the predicate symbols in Neg_P depend is denoted by Neg_P^* . For convenience, we will denote this set simply by N .
- (v) We define P^- to be the set of clauses in P which contain a predicate symbol from N in the head.

Next, we reproduce the definition of acceptability as given in [3].

Definition 3.2 Let P be a normal logic program, let l be a level mapping for P , and let I be a model for P whose restriction to the predicate symbols in N is a model⁶ of the Clark-completion $\text{comp}(P^-)$ of P^- . Then P is called *acceptable with respect to l and I* if the following condition (*) is satisfied: for every clause $A \leftarrow L_1, \dots, L_n$ in $\text{ground}(P)$, the following implication holds for all $i \in \{1, \dots, n\}$:

$$\text{if } I \models \bigwedge_{j=1}^{i-1} L_j \text{ then } l(A) > l(L_i).$$

A program is called *acceptable* if it is acceptable with respect to some level mapping and some model.

It is important to notice that I need not be a Herbrand model here (even though the level mapping is defined on B_P) and, in fact, even the notion of $\text{ground}(P)$ used in Definition 3.2 depends on the choice of preinterpretation in that it depends on the choice of first order language. We will therefore find the following definition useful.

⁶ This requirement implicitly assumes that $\text{comp}(P^-)$ is consistent. In fact, as was shown in [3, Theorem 4.17], this is the case for left-terminating non-floundering programs.

Definition 3.3 A program P which is acceptable with respect to some J -model M (so that the underlying preinterpretation of M is J) will be called *J -acceptable*. (Thus, $\text{ground}(P)$ used in Definition 3.2 becomes $\text{ground}_J(P)$ in our notation, the level mapping l is taken to be defined on $B_{P,J}$ and the condition $(*)$ is required to hold for each J -ground clause in P). If J is a Herbrand-preinterpretation whose associated language contains only the function and constant symbols occurring in the program itself, then P will be called *Herbrand-acceptable*.

The following theorem was established in [3]. A program is called *left-terminating* if all SLDNF-derivations starting in a ground goal are finite, provided the left-most selection rule is used. If the selected literal in some step of an SLDNF-derivation of some goal is non-ground and negative, then the program is said to *flounder* under this goal.

Theorem 3.4 *Every acceptable program is left-terminating. Every left-terminating non-floundering program is Herbrand-acceptable.*

Consequently, if a program is acceptable but not Herbrand-acceptable, it must flounder on some ground query. Since this is not desirable for practical programming purposes under Prolog, Herbrand-acceptability suffices for Prolog programming in the main. Other approaches to handling negation, however, like Chan's constructive negation ([8]), can deal with non-ground negated atoms in the selected literal. Therefore, the notion of acceptability is still important in this context, see [15], and consequently acceptability under arbitrary preinterpretations is worth studying.

Next, we will give a short account of the difficulties involved, by means of some example programs. These programs are not of practical use, and flounder on some inputs. They are designed for the sole purpose of highlighting the importance of choosing suitable preinterpretations.

Program 3.5 Let P_1 be the following program.

$$\begin{aligned} r(0) &\leftarrow \neg p(0), \neg r(0) \\ p(0) &\leftarrow \neg q(X) \\ q(0) &\leftarrow \end{aligned}$$

The program P_1 is acceptable with respect to the model $M = \{p(0), q(0)\}$ whose domain of preinterpretation⁷ is the set $\{0, 1\}$. It is not Herbrand-acceptable, though.

Program 3.6 Let P_2 be the following program.

$$\begin{aligned} r(0) &\leftarrow \neg q(X), \neg r(0) \\ q(0) &\leftarrow \end{aligned}$$

⁷ If not mentioned otherwise, the preinterpretation is supposed to be the identity function on the set of all (ground) terms over the language, which in this case coincides with the domain of preinterpretation.

The program P_2 is Herbrand-acceptable (with respect to the model $\{q(0)\}$) although it flounders on the goal $\{\leftarrow r(0)\}$. However, it has no supported model with respect to the set $\{0, 1\}$ as domain of preinterpretation. We see from this example that choosing a language which contains more than the function and constant symbols occurring in the program can lead to failure in finding a model under this preinterpretation which can be used for determining acceptability.

Constructive negation in the sense of [8] (see also [15]), as a way to resolve floundering, does not cover the general case either, due to the following two assumptions made in the cited papers: Chan in [8, p. 113] assumes, throughout, the consistency of the completed database, and also assumes [8, p. 116] that the underlying language contains infinitely many constant symbols and function symbols. Consistency of the completed database is dependent on the chosen domain of preinterpretation (restricted through the presence of infinitely many constant and function symbols) and, in fact, under this assumption concerning the underlying language, we see that the completed database for program P_2 is not consistent.

Furthermore, consider the following program.

Program 3.7 Let P_3 be the program

$$\begin{aligned} r(0) &\leftarrow \neg q(X), r(0) \\ q(0) &\leftarrow \end{aligned}$$

For program P_3 , the unique supported Herbrand model $\{q(0)\}$ is certainly the desired model⁸. The program is also Herbrand-acceptable with respect to this model.

However, the goal $\leftarrow r(0)$ does not terminate under Chan's constructive negation⁹. In [15], however, it was shown that the set of all programs which are J -acceptable with respect to some preinterpretation J whose domain contains infinitely many constants and functions, coincides with the set of all programs which terminate under Chan's constructive negation. Nevertheless, this result does not account for programs which are, for example, Herbrand-acceptable but not J -acceptable, where J is constrained as above. The Program P_3 displays this fact.

In programs P_1 and P_3 , the Herbrand preinterpretation was too *small* to allow determination of acceptability. Our final program shows that in some cases it may even be too *large*. It also displays the fact that not only the choice of underlying language is important, but also how function symbols are treated by the preinterpretation.

Program 3.8 Let P_4 be the following program.

$$r(0) \leftarrow \neg q(X), r(0)$$

⁸ Different standard semantics, e.g. minimal model semantics, perfect model semantics, stable model semantics etc. coincide with this model in this case.

⁹ Although this goal is bounded, see Section 7.

$q(f(0)) \leftarrow$

Under the Herbrand-preinterpretation, this program is not acceptable due to the existence of the function symbol f , giving an instance of $q(X)$ which is false. However, P_4 is acceptable with respect to a preinterpretation whose domain is the one-point set $\{0\}$ and where f is interpreted as the identity function on $\{0\}$.

We see from the examples above that the choice of a suitable preinterpretation is essential when studying acceptability. In the sequel, we will assume that such a choice has been made, and emphasize this by using the prefix J -as described in the preliminaries in Section 2.

4 J -Acceptable Programs and their Unique Supported J -Models

In [3] it was shown that every Herbrand-acceptable program has a unique supported Herbrand model which can be obtained using iterates of Fitting's three-valued operator Φ_P . We generalize this result and show that every J -acceptable program has a unique supported J -model, which can be obtained as the limit in Q of a sequence of iterates of the single-step operator.

The following generalizes [3, Lemma 6.2].

Lemma 4.1 *Let P be a normal logic program, let J be a preinterpretation for P , let $I \in I_{P,J}$ and let K be a partial J -interpretation for P with $K^+ \subseteq I \subseteq {}^c K^-$. Then $\Phi_{P,J}(K)^+ \subseteq T_{P,J}(I) \subseteq {}^c \Phi_{P,J}(K)^-$. Furthermore, if $K^+ = I = {}^c K^-$, so that K is total, then $\Phi_{P,J}(K)^+ = T_{P,J}(I) = {}^c \Phi_{P,J}(K)^-$.*

Proof. Let $A \in \Phi_{P,J}(K)^+$. Then A must be the head of a clause $A \leftarrow A_1, \dots, A_{k_1}, \neg B_1, \dots, \neg B_{k_2}$ in $\text{ground}_J(P)$ with $A_i \in K^+$ and $B_j \in K^-$ for all $i = 1, \dots, k_1$ and $j = 1, \dots, k_2$. By assumption, it follows that for these values of i and j , $A_i \in I$ and $B_j \notin I$, and hence $A \in T_{P,J}(I)$.

For the second inclusion, it suffices to show that $\Phi_{P,J}(K)^- \subseteq {}^c T_{P,J}(I)$. Let $A \in \Phi_{P,J}(K)^-$. Then, for every clause $A \leftarrow A_1, \dots, A_{k_1}, \neg B_1, \dots, \neg B_{k_2}$ in $\text{ground}_J(P)$, we have some $A_i \in K^-$ or some $B_j \in K^+$. Hence, for every such clause, we have some $A_i \notin I$ or some $B_j \in I$, which implies that $A \notin T_{P,J}(I)$. For the last statement, it suffices to note that a conjunction $L_1 \wedge \dots \wedge L_n$ of literals is true in K if and only if it is true in I if and only if it is not false in K . \square

The following straightforward corollary provides the essential link between the Φ -operator, the single-step operator $T_{P,J}$ and convergence in Q . Intuitively speaking, iterates of $T_{P,J}$ are “squeezed between” the iterates of $\Phi_{P,J}$.

Corollary 4.2 *Let $I_n = T_{P,J}^n(\emptyset)$ and let $K_n = \Phi_{P,J} \uparrow n(\perp)$. Then, for all $n \in \mathbb{N}$, we obtain $K_n^+ \subseteq I_n \subseteq {}^c K_n^-$.*

The following now easily carries over from [3], and is in fact a direct consequence of Lemma 4.1.

Proposition 4.3 *Let P be a normal logic program, let J be a preinterpretation for P , and let $I = (I^+, {}^cI^+)$ be a total J -interpretation for P . Then I is a fixed point of $\Phi_{P,J}$ if and only if I^+ is a fixed point of $T_{P,J}$. Furthermore, if $\Phi_{P,J}$ has exactly one fixed point M and M is total, then M^+ is the unique fixed point of $T_{P,J}$.*

Proof. Let I be a fixed point of $\Phi_{P,J}$. Then $I^+ \subseteq I^+ \subseteq {}^cI^-$ and by Lemma 4.1 we obtain $I^+ = \Phi_{P,J}(I)^+ \subseteq T_{P,J}(I^+) \subseteq {}^c\Phi_{P,J}(I)^- = {}^cI^- = I^+$. Conversely, let I^+ be a fixed point of $T_{P,J}$. By Lemma 4.1, we obtain $\Phi_{P,J}(I)^+ = T_{P,J}(I^+) = I^+ = {}^cI^- = {}^c\Phi_{P,J}(I)^-$, and therefore $\Phi_{P,J}(I)^+ = I^+$ and $\Phi_{P,J}(I)^- = I^-$. The last statement now follows immediately. \square

Collecting together the previous results now yields convergence in Q of iterates of $T_{P,J}$.

Theorem 4.4 *Let P be a normal logic program and let J be a preinterpretation for P . Assume that $M = \Phi_{P,J} \uparrow \omega$ is total. Then $T_{P,J}^n(\emptyset)$ converges in Q to M^+ , and M^+ is the unique supported J -model $M_{P,J}$ of P .*

Proof. Using the notation from Corollary 4.2, we obtain $M^+ = \bigcup K_n^+$ and $M^- = \bigcup K_n^-$. Since M is total, the previous proposition implies that M^+ is the limit in Q of the sequence I_n . Since totality of $\Phi_{P,J} \uparrow \omega$ implies that it is the unique fixed point of $\Phi_{P,J}$, it therefore equals (M^+, M^-) , so that M^+ is the unique fixed point of $T_{P,J}$ by Proposition 4.3. \square

Theorem 4.5 *Let P be a normal logic program and let J be a preinterpretation for P .*

- (1) *If $T_{P,J}(I) = I$, then $\Phi_{P,J} \uparrow \omega \subseteq (I, {}^cI)$.*
- (2) *If P is J -acceptable, then $\Phi_{P,J} \uparrow \omega$ is total.*
- (3) *If P is J -acceptable with respect to some level mapping l , then, for all $n \in \mathbb{N}$ and all $A \in B_{P,J}$ with $l(A) = n$, we have $A \in \Phi_{P,J} \uparrow (n+1)^+ \cup \Phi_{P,J} \uparrow (n+1)^-$, i.e. A is not undefined in $\Phi_{P,J} \uparrow (n+1)$.*

Proof. (1) By Proposition 4.3, $(I, {}^cI)$ is a fixed point of $\Phi_{P,J}$. Since $\Phi_{P,J}$ is monotonic, it has a least fixed point M and $\Phi_{P,J} \uparrow \omega \subseteq M \subseteq (I, {}^cI)$.

(2) The proof given in [3, Theorem 6.7] carries over directly from the Herbrand case, and we omit the details.

(3) This is a direct corollary of the proof of (2). \square

The following main result of this section is now easily obtained.

Corollary 4.6 *Let P be J -acceptable. Then the sequence $(T_{P,J}^n(\emptyset))$ converges in the atomic topology to the unique supported J -model $M_{P,J}$ of P .*

5 The Canonical Level Mapping for J -Acceptable Programs

We show next how to obtain a level mapping for a given program which is suitable for proving its acceptability. The construction is based on a construction of a canonical level mapping for locally hierarchical programs, and it will be convenient to recall it here. For the purpose of the present section, we relax the notion of a level mapping and define a *general level mapping* to be a partial mapping $l : B_{P,J} \rightarrow \alpha$, where α is an arbitrary ordinal; we always extend l to J -instances of literals in the manner defined in Section 3. If we assume that $\alpha = \omega$, the first transfinite ordinal, and that l is total, we obtain our original definition. To avoid confusion, we will refer to a level mapping $l : B_{P,J} \rightarrow \omega$ as an ω -level mapping for the present section.

Definition 5.1 A normal logic program P is called *J -locally hierarchical* or *J -strictly level-decreasing* if there exists a countable ordinal α and level mapping $l : B_{P,J} \rightarrow \alpha$ such that each clause $A \leftarrow L_1, \dots, L_n$ in $\text{ground}_J(P)$ satisfies $l(A) > l(L_i)$ for all $i = 1, \dots, n$.

These programs were introduced by Cavedon in [6] in the context of Herbrand-preinterpretations; the name “strictly level-decreasing” was adopted by us in [19], and will also be used here at times, because this latter name is more indicative of the defining property involved than is the former. These programs are known ([7]) to have a unique supported model which is also their unique perfect model, and it can be obtained by several different methods, including some which are topological in nature, see [18]. Furthermore, the class of all such programs can compute all partial recursive functions, see [18,19].

Every locally hierarchical program has a canonical, i.e. least, level mapping l_P , and we review next the construction from [19].

Construction 5.2 Let P be a normal logic program. We define a general level mapping l_P on $B_{P,J}$ as follows. For every $A \in B_{P,J}$ which does not occur as a head in $\text{ground}_J(P)$, let $l_P(A) = 0$. For every $A \in B_{P,J}$ which occurs as the head of a unit clause but not as the head of any non-unit clause, let $l_P(A) = 0$. Now let $A \in B_{P,J}$ be such that A is the head of some clause(s) in $\text{ground}_J(P)$. Let \mathcal{B}_A be the collection of body-literals occurring in these clauses. Now suppose that for every $B \in \mathcal{B}_A$, $l_P(B)$ is already defined. Let $M_A = \sup_{B \in \mathcal{B}_A} l_P(B)$ and set $l_P(A) = M_A + 1$, if M_A is a successor ordinal, and set $l_P(A) = M_A$, if M_A is a limit ordinal. Then l_P is obtained by transfinitely iterating this procedure. We will refer to l_P , as defined above, as the (*canonical*) *general level mapping* for P .

Proposition 5.3 *Let P be a program which is locally hierarchical with respect to a general level mapping l . Then l_P is a total general level mapping and P is locally hierarchical with respect to l_P .*

Proof. First we show that $\text{dom}(l_P) = B_{P,J}$. Suppose there is $A \in B_{P,J} \setminus$

$\text{dom}(l_P)$; we can further suppose that $l(A)$ is minimal for A with this property. Then there must be some $B \in \mathcal{B}_A$ with $B \notin \text{dom}(l_P)$, otherwise $l_P(A)$ is defined in the process given in Construction 5.2. Since P is strictly level-decreasing with respect to l , we have $l(B) < l(A)$ which contradicts the choice of A with $l(A)$ minimal. Therefore, l_P is a total general level mapping, and obviously P is strictly level-decreasing with respect to it. \square

Proposition 5.4 *Let P be a program which is locally hierarchical with respect to a level mapping l . Then for every $A \in B_{P,J}$, we have $l_P(A) \leq l(A)$.*

Proof. Suppose the conclusion is false. Thus, there is $A \in B_{P,J}$ with $l(A) < l_P(A)$, and such that $l(A)$ is minimal. Then, for all $B \in \mathcal{B}_A$, we have $l(B) < l(A)$ because P is locally hierarchical. Therefore, by minimality of $l(A)$, we have $l(B) \geq l_P(B)$ for all $B \in \mathcal{B}_A$. By definition of l_P , we see that $l_P(A) = \min\{\alpha; \alpha > l_P(B), B \in \mathcal{B}_A\} \leq \min\{\alpha; \alpha > l(B), B \in \mathcal{B}_A\} \leq l(A)$. From this we obtain $l_P(A) \leq l(A)$, giving the required contradiction. \square

We now return to the study of acceptable programs and we will apply Proposition 5.4. For this purpose, let P be a program and let I be a J -model of P . We will now give a program transformation which yields a locally hierarchical program from P and I , allowing us to apply our results on locally hierarchical programs. The program transformation is as follows:

Program Transformation 5.5 Let P be a normal logic program and let I be a J -model of P . For each clause $A \leftarrow L_1, \dots, L_n$ in $\text{ground}_J(P)$ determine the maximal i such that $I \models L_1 \wedge \dots \wedge L_i$. Then replace the given clause with $A \leftarrow L_1, \dots, L_{i+1}$ if $i \neq n$ and by $A \leftarrow L_1, \dots, L_n$ if $i = n$. The resulting J -ground program will be called P_I .

If P is acceptable with respect to I and l , then P_I is locally hierarchical with respect to the ω -level mapping l' (in fact, even acyclic, see [3,6]) which is obtained by restricting l to $B_{P_I,J}$. Therefore, we can obtain the canonical level mapping l_{P_I} of P_I by applying Construction 5.2, and obtain by Proposition 5.3 that l_{P_I} is indeed a total function. Furthermore, by Proposition 5.4 we obtain that $l_{P_I}(A) \leq l'(A)$ for all $A \in B_{P_I,J}$, and since l' maps into ω , the level mapping l_{P_I} also maps into ω . This means, in particular, that Construction 5.2 is in fact not transfinite but closes off at ω .

Definition 5.6 We now define a level mapping l_P for the given program P : for every $A \in B_{P,J} \setminus B_{P_I,J}$, let $l_P(A) = 0$; for every $A \in B_{P_I,J}$, let $l_P(A) = l_{P_I}(A)$.

We summarize the observations that we have just made. We would like to note already that, as a result to be given in Section 6, we will see that a program which is J -acceptable is always J -acceptable with respect to its unique supported J -model $M_{P,J}$.

Construction 5.7 Let P be a normal logic program and let I be a J -model of P .

- (1) Obtain P_I from P and I using Program Transformation 5.5.
- (2) Obtain l_{P_I} from Construction 5.2.
- (3) Obtain l_P from Definition 5.6.

Proposition 5.8 *Let P be J -acceptable with respect to a J -model I . Then the following statements hold.*

- (i) P_I , obtained from step (1) in Construction 5.7, is locally hierarchical.
- (ii) l_{P_I} , obtained from step (2) in Construction 5.7, is total and maps into ω .
- (iii) l_P , obtained from step (3) in Construction 5.7, is total and maps into ω .
- (iv) P satisfies condition (*) of Definition 3.2 with respect to I and l_P .

Proof. It only remains to prove statement (iv), which is immediate from the definition of l_P . \square

In the following, l_P will always denote the (partial) level mapping as given in Construction 5.7. It will be called the *canonical (partial) level mapping* for P with respect to I .

Several modular notions of acceptability have been proposed in the literature, for example *semi-acceptability* in [4], *up-* and *low-acceptability* in [16]. It is in fact not difficult to see that our approach to finding a unique pointwise minimal level mapping can be carried over to these modular notions of acceptability.

6 A Characterization of J -Acceptable Programs

We will first give a minimality result for the unique supported model of an acceptable program and then give our main result, Theorem 6.3. Recall that $M_{P,J}$ denotes the unique supported J -model of P , and it exists if P is J -acceptable.

The following theorem shows that $M_{P,J}$ is suitable for establishing J -acceptability of P , and that $M_{P,J}$ is in fact least with this property. This fact explains the comment of one of the referees of an earlier version of this paper that models which are not minimal are sometimes a better choice for determining acceptability: only one of the minimal models (if any) is suitable for establishing acceptability at all!

Theorem 6.1 *Let P be J -acceptable with respect to some J -model M and level mapping l . Then P is J -acceptable with respect to $M_{P,J}$ and l , and $M_{P,J} \subseteq M$.*

Proof. Let M' be M restricted to the predicate symbols in $N = \text{Neg}_P^*$, let $M'_{P,J}$ be $M_{P,J}$ restricted to the predicate symbols in N and let l' be l restricted

to the predicate symbols in N . Then P^- is J -acceptable with respect to M' and l' . Hence, we obtain that M' is the unique supported J -model of P^- and therefore coincides with $M'_{P,J}$.

Now let P_1 be $P \setminus P^-$, and let **true** and **false** denote atoms which always evaluate to true and false, respectively. Replace every atom with predicate symbol in N in each clause in $\text{ground}_J(P_1)$ by **true** or **false** if this atom is true, respectively false, with respect to M' . The resulting (ground) program P_2 is definite and therefore has a minimal J -model L , which is supported. Since $M_{P,J}$ is the unique supported model of P , we obtain $M_{P,J} = M' \cup L \subseteq M$. By the minimality of L and the fact that P_2 is definite, we obtain that P is indeed acceptable with respect to $M_{P,J}$ and l . \square

The following is the key result in our characterization of acceptability.

Theorem 6.2 *Let P be J -acceptable with respect to I and l . Then P is J -acceptable with respect to $M_{P,J}$ and l_P , where l_P is the canonical level mapping of P with respect to $M_{P,J}$.*

Proof. By Theorem 6.1, P is acceptable with respect to $M_{P,J}$ and l . By Proposition 5.8, P is acceptable with respect to $M_{P,J}$ and l_P . \square

We now collect together our previous results and establish our characterization of J -acceptable programs.

Theorem 6.3 *Let P be a normal logic program. Then P is J -acceptable if and only if the following conditions are satisfied:*

- (i) *The sequence $(T_{P,J}^n(\emptyset))_{n \in \mathbb{N}}$ converges in Q to some J -model M (which is the unique supported J -model $M_{P,J}$ of P).*
- (ii) *The mapping l_P , constructed from P and $I = M$ as in Construction 5.7, is total and takes values in the natural numbers.*
- (iii) *P satisfies $(*)$ from Definition 3.2 with respect to l_P and M .*

As already mentioned in Section 3, it was shown in [15] that the set of all programs which are J -acceptable with respect to some preinterpretation J whose domain contains infinitely many constants and functions, coincides with the set of all programs which terminate under Chan's constructive negation. We therefore note the following corollary to Theorem 6.3.

Corollary 6.4 *A normal logic program P is terminating under Chan's constructive negation if and only if it satisfies conditions (i), (ii) and (iii) of Theorem 6.3 for some preinterpretation J whose domain contains infinitely many constants and functions.*

We will now show that the canonical level mapping l_P of P is least among all level mappings with respect to which acceptability can be established – a result which will be used later in Section 7.

Lemma 6.5 *Let P be J -acceptable with respect to $M_{P,J}$ and a level-mapping l . Then $l_P(A) \leq l(A)$ for all $A \in B_{P,J}$, where l_P is the canonical level mapping of P with respect to $M_{P,J}$.*

Proof. Let P_I be defined as in Section 5. For $A \in B_{P_I,J}$, we obtain $l_P(A) \leq l(A)$ by the minimality property established in Proposition 5.4, since P_I is strictly level-decreasing. If $A \in B_{P,J} \setminus B_{P_I,J}$, then by definition of l_P we have $l_P(A) = 0 \leq l(A)$ as desired. \square

Theorem 6.6 *For any J -acceptable program P , the canonical level mapping l_P with respect to $M_{P,J}$ is least among all level-mappings with respect to which P can be shown to be J -acceptable. More precisely, if P is J -acceptable with respect to some J -model I and some level mapping l , then for all $A \in B_{P,J}$ we have $l_P(A) \leq l(A)$.*

Proof. Let P be J -acceptable with respect to some J -model I and some level mapping l , and let $A \in B_{P,J}$ be arbitrarily chosen. By Theorem 6.1, P is J -acceptable with respect to l and $M_{P,J}$. By Lemma 6.5 we obtain $l_P(A) \leq l(A)$ as desired. \square

7 Termination of Non-Ground Queries

We cite the following result from [1, Theorem 5.7]. For a partial converse, see [3]. Recall that a literal L is called *bounded* with respect to a level mapping l if there exists a natural number n such that $l(L') \leq n$ for all ground instances L' of L .

Theorem 7.1 *Let P be acceptable with respect to a level mapping l and a model I . Then, for every literal L which is bounded with respect to l , all SLDNF-derivations of $P \cup \{\leftarrow L\}$, using the Prolog selection rule, are finite. In particular, the goal $\{\leftarrow L\}$ terminates under Prolog.*

With these preparations, the following result is easily obtained.

Proposition 7.2 *Let P be J -acceptable with respect to a level mapping l and a J -model I , and let L be a literal which is bounded with respect to l . Then L is bounded with respect to l_P .*

Proof. This follows immediately from the minimality of l_P established in Theorem 6.6. \square

We will now discuss termination of non-ground goals. The following notions were introduced in [3].

A *multiset* or *bag* over a set W is an unordered sequence of elements of W . Given a (non-reflexive) ordering $<$ on a set W , the *multiset ordering* over $(W, <)$ is an ordering of finite multisets of the set W and is defined as follows. For two finite multisets X and Y over W , let $X \prec Y$ if and only if $X = (Y \setminus \{a\}) \cup Z$ for some finite multiset Z such that $b < a$ for all $b \in Z$.

Finally, define the multiset ordering over $(W, <)$ as the transitive closure of the relation \prec . The multiset whose elements are a_1, \dots, a_n will be denoted by $\text{bag}(a_1, \dots, a_n)$.

The following definition is to be found in [3, Definition 2.9].

Definition 7.3 Let P be a program, let l be a level mapping for P , let I be a model of P with $I \cap N$ being a model for P^- , and let $k \geq 0$.

(i) With each ground goal G of the form $\leftarrow L_1, \dots, L_n$, we associate a finite multiset $l_I(G)$ of natural numbers, where $l_I(G) = \text{bag}(l(L_1), \dots, l(L_n(G, I)))$ in which $n(G, I) = \min(\{n\} \cup \{i \in \{1, \dots, n\} \mid I \not\models L_i\})$.

(ii) With each goal G , we associate a set of multisets $l'_I(G)$ defined by $l'_I(G) = \{l_I(G') \mid G' \text{ is a ground instance of } G\}$.

(iii) A goal G is called *bounded by k* with respect to l and I if $k \geq j$ for $j \in \bigcup l'_I(G)$, where $\bigcup l'_I(G)$ stands for the set-theoretic union of the elements of $l'_I(G)$.

(iv) A goal is called *bounded* with respect to l and I if it is bounded by some $k \geq 0$ with respect to l and I .

It was observed in [1] that the choice of level mapping and of the model can affect the class of (non-ground) goals whose termination can be established, since the choice of both the level mapping and the model affect the notion of boundedness for goals. However, we will prove that the model $M_{P,J}$ and the canonical level mapping l_P are completely general for proving termination of non-ground goals.

The following result is taken from [3, Corollary 4.11]. A partial converse is also given there.

Theorem 7.4 *Let P be an acceptable program and let G be a bounded goal. Then all SLDNF-derivations of $P \cup \{G\}$ using the Prolog selection rule are finite.*

Our minimality results allow us to establish the following theorem.

Theorem 7.5 *Let P be J -acceptable with respect to a level mapping l and a J -model I , and let G be a goal which is bounded with respect to l and I . Then G is bounded with respect to l_P and $M_{P,J}$.*

Proof. Since $l_P(A) \leq l(A)$ for all $A \in B_{P,J}$ by Theorem 6.6, it suffices to show that $n(G, M_P) \leq n(G, I)$. This, however, follows directly from the minimality property given in Proposition 6.6. \square

8 Conclusions

In the framework of studying unique supported model classes, as mentioned in the Introduction, our results provide a neat characterization of acceptability. Three consecutive conditions have to be satisfied: convergence in Q of iterates of T_P ; totality of l_P ; condition $(*)$ from Definition 3.2. If they hold, the

program is acceptable. If a program is known to have a unique supported (J -)model, then this model is sufficient for establishing (J -)acceptability.

However, even in the Herbrand case, our characterization does not yield a straightforward procedure which could be automated since the determination of $M_{P,J}$ is an undecidable problem (convergence in Q is undecidable). However, we believe that it aids in establishing acceptability proofs “by hand”, and that the theoretical results could simplify further research on acceptability.

It should be noted here that the notion of semi-acceptability, as introduced in [4], can also be treated in our framework, basically because it is a modular notion of acceptability. Work on this and similar topics is in progress by the authors.

Acknowledgement

The authors wish to thank three anonymous referees of an earlier version of this paper, whose comments quite substantially improved the presentation. We also thank two anonymous referees of the present version of this paper for their suggestions which among others led to the remark in the paragraph at the end of Section 5.

References

- [1] Apt, K. R., *Program Verification and Prolog*. In: Börger E., (Ed.), Specification and Validation Methods for Programming Languages and Systems, Oxford University Press, 1995, pp. 55–95.
- [2] Apt, K. R., Blair, H. A., and Walker, A., *Towards a Theory of Declarative Knowledge*. In: Minker J., (Ed.), Foundations of Deductive Databases and Logic Programming. Morgan Kaufmann Publishers Inc., Los Altos, 1988, pp. 89–148.
- [3] Apt, K. R., and Pedreschi, D., *Reasoning about Termination of Pure Prolog Programs*, Information and Computation **106** (1993), 109–157.
- [4] Apt, K. R., and Pedreschi, D., *Modular Termination Proofs for Logic and Pure Prolog Programs*. In: Levi, G., (Ed.), Advances in Logic Programming Theory, Oxford University Press, Oxford, 1994, pp. 183–229.
- [5] Batarekh, A., and Subrahmanian, V. S., *Topological Model Set Deformations in Logic Programming*, Fundamenta Informaticae **12** (3) (1989), 357–400.
- [6] Cavedon, L., *Continuity, Consistency, and Completeness Properties for Logic Programs*. Proceedings of the 6th International Conference on Logic Programming, Lisbon, Portugal, 1989, pp. 571–584.
- [7] Cavedon, L., *Acyclic Logic Programs and the Completeness of SLDNF-Resolution*, Theoretical Computer Science **86** (1991), 81–92.

- [8] Chan, D., *Constructive Negation Based on the Completed Database*. In: Proc. of the 5th Int. Conf. and Symp. on Logic Programming, 1988, pp. 111–125.
- [9] Etalle, S., Bossi, A., and Cocco, N., *Termination of Well-moded Programs*, J. Logic Programming **38** (1999), 243–257.
- [10] Fitting, M., *A Kripke-Kleene Semantics for General Logic Programs*, J. Logic Programming **2** (1985), 295–312.
- [11] Fitting, M., *Metric Methods: Three Examples and a Theorem*, J. Logic Programming **21** (3) (1994), 113–127.
- [12] Hitzler, P., *Topology and Logic Programming Semantics*. Diplomarbeit in Mathematik, Universität Tübingen, 1998.
- [13] Hitzler, P., and Seda, A. K., *Characterizations of Classes of Programs by Three-Valued Operators*. In: Proceedings of the 5th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR'99), El Paso, Texas, December, 1999. Springer Lecture Notes in Artificial Intelligence, pp. 1–15, to appear.
- [14] Lloyd, J. W., *Foundations of Logic Programming*. Second Edition, Springer, Berlin, 1988.
- [15] Marchiori, E., *On Termination of General Logic Programs with respect to Constructive Negation*, J. Logic Programming **26** (1) (1996), 69–89.
- [16] Marchiori, E., *A Methodology for Proving Termination of General Logic Programs*. Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI'95), 1995, 356–367.
- [17] Seda, A. K., *Topology and the Semantics of Logic Programs*, Fundamenta Informaticae **24** (4) (1995), 359–386.
- [18] Seda, A. K., and Hitzler, P., *Topology and Iterates in Computational Logic*. Proceedings of the 12th Summer Conference on Topology and its Applications: Special Session on Topology in Computer Science, Ontario, August 1997. Topology Proceedings **22**, Summer 1997, 427–469.
- [19] Seda, A. K., and Hitzler, P., *Strictly Level-decreasing Logic Programs*. In: Butterfield, A., and Flynn, S., (Eds.), Proceedings of the Second Irish Workshop on Formal Methods (IWF'98), Electronic Workshops in Computing, British Computer Society, 1999, pp. 1–18.
- [20] Willard, S., *General Topology*. Addison Wesley Series in Mathematics, Addison Wesley Publishing Company Inc., Massachusetts, 1970.