Fields Institute Communications
Volume **00**, 0000

The Fixed-Point Theorems of Priess-Crampe and Ribenboim in Logic Programming

Pascal Hitzler
Department of Mathematics
University College Cork, Cork, Ireland
phitzler@ucc.ie

Anthony Karel Seda
Department of Mathematics
University College Cork, Cork, Ireland
aks@ucc.ie

Abstract. Sibylla Priess-Crampe and Paulo Ribenboim recently established a general fixed-point theorem for multivalued mappings defined on generalized ultrametric spaces, and introduced it to the area of logic programming semantics. We discuss, in this context, the applications which have been made so far of this theorem and of its corollaries. In particular, we will relate these results to Scott-Ershov domains, familiar in programming language semantics, and to the generalized metrics of Khamsi, Kreinovich and Misane which have been applied, by these latter authors, to logic programming. Amongst other things, we will also show that a unified treatment of the fixed-point theory of wide classes of programs can be given by means of the theorems of Priess-Crampe and Ribenboim.

1 Introduction

In simple terms, logic programming is concerned with the use of logic as a programming language. What this means in practice is making deductions from sets of clauses, or rules, by means of an interpreter or automated theorem prover. The reference [3] is an excellent account of the growth of logic programming and of its current status as a major tool in various parts of computer science, such as database systems, artificial intelligence, natural language processing, machine learning and building expert systems etc. For most of its first twenty five years of existence, much of the emphasis in logic programming, or more particularly on its implementation

1991 *Mathematics Subject Classification.* 54H25, 68N17, 68Q55.

The first named author acknowledges financial support under grant SC/98/621 from Enterprise Ireland. The second named author acknowledges very substantial support from the University of Saskatchewan in Saskatoon in presenting this paper at the International Conference and Workshop on Valuation Theory.

via Prolog, has been on its capabilities as a stand-alone programming language, and on comparing its advantages and disadvantages over conventional imperative and object-oriented programming languages such as C, C^{++} and Java. Today, logic programming is entering an exciting phase of growth, maturing into the broader subject of computational logic. This latter term is often interpreted to mean the use of logic generally within computer science, but also refers to the growing trend in large systems and industrial applications of using logic programming languages in partnership with languages such as Java, with different tasks being undertaken in different languages and interfaces provided between them, see [3] for illustrations of these developments.

A logic programming system comprises four main facets: (i) the syntax or expressiveness of the system and its computational adequacy (the ability, or otherwise, to compute all partial recursive functions); (ii) the procedural semantics of the system or, in other words, what is output by the interpreter; (iii) the declarative semantics or logical meaning of the output; (iv) the fixed-point semantics. These four issues are, of course, highly interconnected and, in particular, it is important that the three semantics mentioned coincide in some sense. In fact, what is meant by the "declarative semantics" is usually some natural model canonically associated with each program permitted by the syntax, and realized as the (least, minimal, unique etc.) fixed point of an operator determined by the program. Unfortunately, most systems with enhanced syntax permit many canonical models, and it is by no means clear in general which of them best captures the intended meaning of the programmer. Indeed, the study of these standard models, such as the well-founded model [10], the stable model [11] or the perfect model [24], and of the corresponding operators, accounts for a high proportion of the research undertaken on the foundations of the subject.

In conventional programming language semantics, such as the denotational semantics of imperative programs, fixed points of operators (and of functors) play an important rôle, and indeed are fundamental wherever recursion and self-reference are encountered. However, in that context the operators which arise are usually monotonic, indeed continuous. Therefore, apart from some applications of the Banach contraction mapping theorem in concurrency, and attempts to unite the order-theoretic and metric approaches (see [26, 31]), the main fixed-point theorem in general use in classical semantics is the well-known Knaster-Tarski theorem which we state in the following form: *if T is defined and monotonic on a complete partial order X, then T has a least fixed point which is also the least pre-fixed point of T.* In fact, if T is continuous, then the least fixed point of T is the supremum of the set of iterates $T^n(\perp)$, where $\perp$ denotes the bottom element of X, see [32]. Furthermore, the same representation of the least fixed point of T can even be obtained for arbitrary monotonic T if one works transfinitely with ordinal powers, see [19]. On the other hand, the situation in the semantics of logic programs is very different. Once one introduces negation, which is certainly implied by the term "enhanced syntax" used earlier, then certain of the important operators are not monotonic (and therefore not continuous), and in consequence the Knaster-Tarski theorem is no longer applicable to them. Various ways have been proposed to overcome this problem. One such is to introduce syntactic conditions on programs, see [1, 24] for example, and to disallow those programs not meeting these conditions, in an attempt to recover continuity in the order-theoretic sense. Another is to consider different operators, and we discuss this later. The third main solution is to introduce techniques from topology and analysis to augment arguments based on order, and it is this theme which will mainly concern us in this paper, see [5] for a discussion of the rôle of "continuous" mathematics in this context. Thus, one finds methods based on topology and dynamical systems ([4, 13, 14, 16, 27, 29, 30]), methods based on metrics ([9, 12, 17]), methods based on quasi-metrics ([15, 28]) and finally, one finds methods based on ultrametric spaces.

In fact, metric and ultrametric methods were introduced to logic programming by Fitting in [9], although all the metrics he considered are actually ultrametrics as is usually the case in computing, see [31]. Nevertheless, they take their values in the real numbers, and thus the fixed-point theorem that is applied in [9] is the Banach contraction mapping theorem[1]. This is rather restrictive in so much as it is often useful to make transfinite constructions and definitions, although these may well be shown later to close off at ω as this is important for computability purposes. A much more general approach is provided in [6] and in the papers [20, 21, 22] of Priess-Crampe and Ribenboim. These papers are concerned with ultrametric spaces whose distance functions take their values in an arbitrary partially ordered set, not just in the real numbers, together with natural generalizations of the Banach theorem for both single-valued and multivalued mappings. This setting offers a highly flexible framework in which to study the fixed-point theory necessary for logic programming semantics, and it is the purpose of the present paper to discuss this point of view in relationship to conventional methods. In particular, we will focus mainly on two of the standard models mentioned earlier, the perfect model and the stable model, and show how one can utilize the fixed-point theorems of Priess-Crampe and Ribenboim [20, 21] to establish their existence and certain of their basic properties in a rather simple and natural way. Potential applications also exist to the various other standard models and to systems involving uncertainty, but these remain to be fully investigated and will not be discussed here in detail. Nevertheless, it should become clear that the fixed-point theorems of [20, 21] are, in many cases, a viable and important alternative to the Knaster-Tarski theorem in situations where this latter result is not available. Indeed, we give a strong hint in Section 4 that these theorems are likely to have applications in the wider context of theoretical computer science. Moreover, even where the Knaster-Tarski theorem is applicable, it does not provide conditions under which the fixed point can be seen to be unique. This extra information, which may be important, can sometimes be settled by the fixed-point theorems of Priess-Crampe and Ribenboim, and we give an illustration of this in Section 4.

In the main, this paper is expository with references given to original sources, although some new results are presented. Thus, the overall plan of the paper is as follows. In Section 2, we present the minimum amount of logic programming theory that is needed to make the work relatively self-contained and intelligible to the non-specialist, and at the same time establish the notation we use. In Section 3, we will record the basic facts concerning generalized ultrametric spaces and the fixed-point theorems of Priess-Crampe and Ribenboim which we wish to

---

[1] One of the main applications Fitting makes in [9] is to the acceptable programs of Apt and Pedreschi, see [2], which arise in the context of termination proofs. In fact, there is an error in [9] to the extent that the (ultra)metric d_3 defined there is not in fact a metric since it fails to satisfy the axiom $d(x,x) = 0$, although it satisfies the other usual axioms; it is therefore an example of the *weak metrics* defined in [16]. Indeed, the problem just mentioned was overcome in [16] by establishing a version of the Banach theorem applicable to weak metrics.

apply. Sections 4 and 5 contain two applications of these theorems in the area of logic programming semantics. The first is based on a general method of casting a Scott-Ershov domain into a generalized ultrametric space. This will enable us to show, by applying the fixed-point theorems, that every locally stratified program has a supported model, and that every locally hierarchical program has a unique supported model (its perfect model). The second application we make is to the stable model semantics of disjunctive programs and databases. In fact, we show finally in Section 5 by means of the multivalued fixed-point theorem of [21] that every locally stratified extended disjunctive logic program (or database) admits a stable model. Thus, in this manner, we obtain a unified approach to the fixed-point theory of the (very general) class of locally stratified programs and databases. It is possible that the same methods will unify the fixed-point theory of other classes of programs, and this is a question which is under investigation.

Acknowledgement We thank Professors Sibylla Priess-Crampe and Paulo Ribenboim for keeping us informed of their work on ultrametric spaces, and for encouraging us to write this account of its applications to logic programming. We also thank an anonymous referee for making helpful comments which improved the presentation of this work in several places.

2 Normal Logic Programs

2.1 Syntax of Normal Logic Programs. Given a first-order language $\mathcal{L}$, a *normal logic program P with underlying language $\mathcal{L}$* is a finite set of *clauses* of the form

$$\forall(A \leftarrow B_1 \wedge \cdots \wedge B_n \wedge \neg B_{n+1} \wedge \neg B_m),$$

usually written as

$$A \leftarrow B_1, \ldots, B_n, \neg B_{n+1}, \ldots, \neg B_m,$$

or more simply as $A \leftarrow \texttt{body}$, where A and all the B_i are atoms in $\mathcal{L}$, $\leftarrow$ denotes implication and the universal quantifier is understood. The atom A is called the *head* of the clause and $B_1, \ldots, B_n, \neg B_{n+1}, \ldots, \neg B_m$ is called the *body* of the clause. By an abuse of notation, we allow m to be zero or, in other words, we allow the body to be empty, in which case we are dealing with the *unit clause*, or *fact*, $A \leftarrow$. A program P is called *positive* or *definite* if no clause contains a negated atom. Our standard reference to concepts in logic programming is [19], and we give next an interesting example of a normal logic program.

Example 2.1 *The following program (taken from* [2]*) computes the transitive closure of a graph.*

$$\begin{array}{rcl}
r(X,Y,E,V) & \leftarrow & m([X,Y],E) \\
r(X,Z,E,V) & \leftarrow & m([X,Y],E), \neg m(Y,V), r(Y,Z,E,[Y|V]) \\
m(X,[X|T]) & \leftarrow & \\
m(X,[Y|T]) & \leftarrow & m(X,T) \\
e(a) & \leftarrow & \qquad \textit{for all } a \in N
\end{array}$$

Here, N denotes a finite set containing the nodes appearing in the graph as elements. In the program, uppercase letters denote variable symbols, lowercase letters constant symbols, and lists are written using square brackets as usual under Prolog. One evaluates a goal (the negation of the object one wishes to compute) such as $\leftarrow r(x,y,e,[x])$, where x and y are nodes and e is a graph specified by a list of pairs *denoting its edges. The goal is supposed to succeed (i.e. the interpreter outputs "yes") when x and y can be connected by a path in the graph. The predicate m implements membership of a list. The last argument of the predicate r acts as an accumulator which collects the list of nodes which have already been visited in an attempt to reach y from x.*

The standard way of implementing interpreters for such programs is via a form of resolution known as SLD-resolution. In this paradigm, one uses first-order predicate logic as a knowledge representation language to specify the problem, and thus a program is a first-order theory. Computation is then viewed as deduction from the program statements (the axioms). It is a standard fact (the computational adequacy theorem) that any partial recursive function (computable function) can be computed by a program (even a definite program) of the type we are discussing relative to SLD-resolution, and hence logic programming systems have the same power as any of the conventional imperative programming languages. In a good implementation, they also have comparable speed in processing and execution, and much shorter code which is closer to the specification of the problem than is the case with procedural languages. However, such issues fall under the heading of "procedural semantics", whereas our main concern here is with the "declarative", or logical semantics, and with the fixed-point semantics of logic programs.

2.2 Semantics of Normal Logic Programs. The usual approach to the declarative semantics of logic programs P is via Tarski's notions of interpretation and model, which are standard apparatus in mathematical logic. However, since we are at all times dealing with sets of clauses, Herbrand interpretations will suffice. Thus, given a logic program P with underlying language $\mathcal{L}$, we form the *Herbrand base* B_P of all ground (variable-free) atoms in $\mathcal{L}$. Then an *interpretation* or *valuation* for P is simply a mapping from B_P to the classical, or two-valued, truth set $\{\mathit{true}, \mathit{false}\}$. Such an interpretation gives a truth value to each ground atom in $\mathcal{L}$ and extends, in the usual way, to give truth value to any closed well-formed formula, including clauses. Moreover, each interpretation can be identified with the subset of B_P on which it takes the value *true*. Thus, the set I_P of all interpretations will be naturally identified with the power set of B_P; it therefore carries the structure of a complete lattice (and hence a complete partial order) under the order of set inclusion. In particular, a *model* for P is an interpretation I for P such that all clauses in P evaluate to true in I. Of course, models are of particular importance in studying the semantics of P. Since clauses are universally quantified, checking their truth amounts to checking the truth of all their ground instances. We denote the set of all ground instances of clauses in P by ground(P), and it is often this set that one works with, rather than with P, when discussing questions of a theoretical nature.

A *partial interpretation* or *three-valued interpretation* I is a mapping from B_P to the truth set $\{\mathit{true}(t), \mathit{false}(f), \mathit{undefined}(u)\}$ and can be identified with a pair (I^+, I^-) of disjoint subsets of B_P. Given a partial interpretation $I = (I^+, I^-)$, atoms in I^+ carry the truth value *true* in I and atoms in I^- the value *false* in I. Atoms which are neither in I^+ nor in I^- carry the truth value *undefined*. Partial interpretations are interpreted in one of the standard three-valued logics such as Kleene's strong three-valued logic which tells one how the undefined value, u, relates to the other truth values under conjunction, disjunction and negation, see [8, 13, 14]. Once this is done, a truth value can be given to any ground formula in $\mathcal{L}$. A partial interpretation (I^+, I^-) is called *total* if $I^+ \cup I^- = B_P$, and such an interpretation can be naturally identified with an element of I_P. The set $I_{P,3}$ of all partial 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 $\bot = (\emptyset, \emptyset)$. Total interpretations are in fact maximal elements in the given ordering.

2.3 Operators Associated with Programs. There are various important operators associated with a logic program P which map interpretations to interpretations. We discuss two of them now and others later on. The first, and perhaps the most important, is the *single-step operator* $T_P : I_P \to I_P$ due to Kowalski and van Emden and defined by letting $T_P(I)$ denote the set of all $A \in B_P$ such that there exists a clause $A \leftarrow \texttt{body}$ in ground(P) with `body` being true in I (in classical two-valued logic).

The operator T_P has many important properties, and we summarize some of these next. First, if P is definite, then T_P is continuous on the complete lattice I_P. Therefore, it has a least fixed point $\text{lfp}(T_P)$ given by the Knaster-Tarski theorem. Moreover, one has the following theorem due to Apt, Kowalski and van Emden which, amongst other things, gives a form of the Gödel completeness theorem relating soundness and completeness for definite logic programming systems.

Theorem 2.2 *For any definite logic program P, we have* $\text{lfp}(T_P) = T_P \uparrow \omega = \{A \in B_P \mid P \vdash A\} = \{A \in B_P \mid P \models A\} = M_P$.

Thus, provability ($\vdash$) from P of a ground atom relative to SLD-resolution coincides with it being a logical consequence ($\models$) of P, and both coincide with truth relative to the *least Herbrand model* M_P, which is the intersection of all Herbrand models of P. Moreover, because of continuity, the iterates T_P^n of T_P close off at ω, which gives us the means, in principle, of finding M_P. For these reasons, M_P is, for definite programs P, usually taken to be the standard model of P or, in other words, the programmer's intended meaning of P as mentioned in the Introduction.

Next, for any normal logic program, whether definite or not, T_P has the pleasing property that an interpretation I is a (two-valued) model of P iff $T_P(I) \subseteq I$ or, in other words, iff I is a prefixed point of T_P. The fixed points of T_P are of particular importance since they are the models of the Clark completion of P which we will not discuss here, but see [19] for details. Moreover, the fixed points of T_P are also called the *supported* models of P. It is strongly argued in [1] that they are the appropriate models to consider, since an atom A belongs to such a model M iff there is a clause $A \leftarrow \texttt{body}$ in ground(P) with `body` true in M, and hence the program itself supports the belief that A is true in M. In particular, this should apply to any model viewed as a candidate for "the" standard model of P, that is, the one best able to capture the intended meaning of P.

Thus, the fixed points of T_P are fundamental in studying the semantics of logic programming systems. Yet a major problem arises: if P is not definite, then T_P is not monotonic as can easily be seen by considering the program with the two clauses $p(0) \leftarrow$ and $p(s(x)) \leftarrow \neg p(x)$ which computes the even natural numbers. Therefore, the Knaster-Tarski theorem is not in general applicable as a means of finding fixed points.

The second operator we consider, but only briefly, is due to Fitting [8] and is the three-valued operator Φ_P defined as a mapping on partial interpretations $K = (K^+, K^-)$ as follows. We set $\Phi_P(K) = (I^+, I^-)$, where I^+ is the set of all $A \in B_P$ with the property that there exists a clause $A \leftarrow \texttt{body}$ in ground(P) such that `body` is true in K, and I^- is the set of all $A \in B_P$ such that for all clauses $A \leftarrow \texttt{body}$ in ground(P) we have that `body` is false in K; truth and falsehood being taken here relative to a three-valued logic as mentioned earlier.

We note that Φ_P is always monotonic but not necessarily continuous. Thus, the Knaster-Tarski theorem applies and shows the existence of a least fixed point of Φ_P, although we may really have to iterate into the transfinite to reach it in the absence of continuity. It was shown in [8], and in [2, 13, 14] for acceptable programs, how fixed points of Φ_P relate to those of T_P. We see later in Section 4 that the fixed-point theorems of [20, 21] can sometimes be applied to show uniqueness of the fixed points of Φ_P.

3 The Fixed-Point Theorems of Priess-Crampe and Ribenboim

It will be convenient to begin this section by giving some basic definitions, introducing some notation and stating the theorems of Priess-Crampe and Ribenboim in the precise form we want for the later applications; all of this is to be found in [20, 21].

Definition 3.1 *Let X be a set and let Γ be a partially ordered set with least element 0. The pair (X, d) is called a* generalized ultrametric space *or simply an* ultrametric space *if $d : X \times X \to \Gamma$ is a function satisfying the following conditions for all $x, y, z \in X$ and $\gamma \in \Gamma$:*
(1) $d(x, y) = 0$ if and only if $x = y$;
(2) $d(x, y) = d(y, x)$;
(3) if $d(x, y) \le \gamma$ and $d(y, z) \le \gamma$, then $d(x, z) \le \gamma$.
For $0 \ne \gamma \in \Gamma$ and $x \in X$, the set $B_\gamma(x) = \{y \in X \mid d(x, y) \le \gamma\}$ is called a γ-ball *or just a* ball *in X with* centre x *and* radius γ.

A substitute is needed in the present context for the usual notion of completeness in (ultra)metric spaces, and this is provided by the notion of "spherical completeness" as follows. An ultrametric space X is called *spherically complete* if $\bigcap \mathcal{C} \ne \emptyset$ for any chain[2] $\mathcal{C}$ of balls in X.

We will be concerned at certain places with fixed points of multivalued mappings, that is, with mappings $f : X \to 2^X$, where 2^X denotes the power set of a set X. A *fixed point* of such a mapping f is a point $x \in X$ such that $x \in f(x)$. A multivalued mapping f is called *non-empty* if, for all $x \in X$, $f(x) \ne \emptyset$.

Whilst the standard notion of contraction involving a numerical constant $k < 1$ is not available in this context, appropriate and useful contractivity notions for mappings defined on ultrametric spaces can be given as follows.

Definition 3.2 *A multivalued mapping $f : X \to 2^X$ on an ultrametric space X is called*
(i) contracting *if for all $x, y \in X$ and for every $a \in f(x)$ there exists an element $b \in f(y)$ such that $d(a, b) \le d(x, y)$.*
(ii) strictly contracting *if for all $x, y \in X$ with $x \ne y$ and for every $a \in f(x)$ there exists $b \in f(y)$ such that $d(a, b) < d(x, y)$.*

[2] By a "chain of balls" we mean, of course, a set of balls which is totally ordered by inclusion.

(iii) strictly contracting on orbits *if for all $x \in X$ and for every $a \in f(x)$ with $a \neq x$ there exists $b \in f(a)$ such that $d(b,a) < d(a,x)$.*

For a multivalued mapping $f : X \to 2^X$, let $\Pi_x = \{d(x,y) \mid y \in f(x)\}$, and for a subset $\Delta \subseteq \Gamma$ denote by $\min \Delta$ the set of all minimal elements of Δ.

The central theorem we need is as follows.

Theorem 3.3 (Priess-Crampe and Ribenboim) [21, (3.1)] *Let (X,d) be a spherically complete ultrametric space. Let $f : X \to 2^X$ be a non-empty contraction which is strictly contracting on orbits, and assume that for every $x \in X$ the set $\min \Pi_x$ is finite and that every element of Π_x has a lower bound in $\min \Pi_x$. Then f has a fixed point.*

This result has several corollaries, both for multivalued mappings and for single-valued mappings, and we state next those that we need in the sequel.

Theorem 3.4 (Priess-Crampe and Ribenboim) [21, (3.4)] *Let (X,d) be spherically complete and let Γ be narrow, that is, such that every trivially ordered subset of Γ is finite. Let $f : X \to 2^X$ be non-empty, strictly contracting on orbits and such that $f(x)$ is spherically complete for every $x \in X$. Then f has a fixed point.*

Theorem 3.5 (Priess-Crampe and Ribenboim) [6, 21] *Let (X,d) be an ultrametric space which is spherically complete, and let $f : X \to X$ be contracting. Then either f has a fixed point or there exists a ball $B_\pi(z)$ such that $d(y, f(y)) = \pi$ for all $y \in B_\pi(z)$. If, in addition, f is strictly contracting on orbits, then f has a fixed point. Finally, this fixed point is unique if f is strictly contracting on X.*

We are now in a position to discuss the rôle of these theorems in the context of logic programming semantics and we proceed to do this next.

4 Locally Stratified Programs

Let P be a normal logic program. A *level mapping* l for P is simply a mapping $l : B_P \to \gamma$, where γ denotes an arbitrary countable ordinal. In fact, γ will be regarded as the set of all ordinals n such that $n \in \gamma$, that is, the set of ordinals n such that $n < \gamma$. Level mappings have been used in logic programming in a variety of contexts including problems concerned with termination, and with completeness, and also to define (generalized) metrics, see [2, 9, 28, 31]. We will shortly see how they can be used to define ultrametrics in the sense of Definition 3.1. However, one of their main uses is in providing syntactic conditions on programs involving negation under which a satisfactory standard model can be obtained. This is often done by using the level mapping as a syntactic device to prevent "negation through recursion", that is, to prevent an atom occurring in the head of a clause and simultaneously occurring negated in its body. This idea is illustrated by the following important definition.

Definition 4.1 *Let P be a normal logic program, let $l : B_P \to \gamma$ be a level mapping and let $A \leftarrow A_1, \dots, A_{k_1}, \neg B_1, \dots, \neg B_{l_1}$ denote a typical clause in* ground(P). *Then P is called:*
(1) locally stratified (with respect to l) *if the inequalities $l(A) \geq l(A_i)$ and $l(A) > l(B_j)$ hold for all i and j in each clause in* ground(P).
(2) locally hierarchical (with respect to l) *if the inequalities $l(A) > l(A_i), l(B_j)$ hold for all i and j in each clause in* ground(P).

The locally stratified programs form one of the most important classes in logic programming and were introduced by Przymusinski in [24]; these programs are in fact a generalization of the stratified programs defined by Apt, Blair and Walker in [1]. Przymusinski gave a non-constructive, and fairly involved, argument to show that each locally stratified program has a unique natural, supported model, known as the perfect model, preferable to any other model in a precise sense defined in [24]; constructive proofs of its existence and properties were given in [7, 29]. Of course, the locally hierarchical programs form a strict subclass of the locally stratified programs. Furthermore, it is known that many programs used in practice fall into the former class (of locally hierarchical programs), that each program in it has a unique supported model ([7, 29]) and that this class is computationally adequate provided that the safe use of cuts is allowed ([30]). We will show next how the fixed-point theory of these classes of programs can be treated by means of the theorems in Section 3. In fact, this was carried out in [29] under a restriction which we are now able to remove. Therefore, we will suppress details of proof and refer to [29] except where the results of [29] are being generalized.

4.1 Domains as Ultrametric Spaces. It is our intention here to cast I_P and $I_{P,3}$ into ultrametric spaces. As we show next, this construction can be carried out in the very general context of a domain and applied to the case of spaces of interpretations. Domains were introduced independently by D.S. Scott and Y.L. Ershov as a means of providing structures for modelling computation, and to provide spaces to support the denotational semantics approach to understanding programming languages, see [32]. Usually, domains are endowed with the Scott topology, which is one of the T_0 (but not T_1) topologies of interest in theoretical computer science. However, as we will see, domains can be endowed with the structure of a spherically complete ultrametric space. This is not something normally considered in domain theory. However, given that there are many ultrametrics which are useful in theoretical computer science, see [31] for some examples, it suggests that a study of the properties of generalized ultrametric spaces, as carried out in [18, 25] and in the papers of Priess-Crampe and Ribenboim, from this viewpoint is worthy of consideration.

Definition 4.2 *A partially ordered set* $(D, \sqsubseteq)$ *is called a* Scott-Ershov domain *with set* D_C *of compact elements (see* [32]*), if the following conditions hold:*
(i) $(D, \sqsubseteq)$ *is a complete partial order (cpo), that is,* D *has a bottom element* $\bot$*, and the supremum* $\sup A$ *exists for all directed subsets* A *of* D*.*
(ii) The elements $a \in D_C$ *are characterized as follows: whenever* A *is directed and* $a \sqsubseteq \sup A$*, then* $a \sqsubseteq x$ *for some* $x \in A$*.*
(iii) For each $x \in D$*, the set* $\mathrm{approx}(x) = \{a \in D_C; a \sqsubseteq x\}$ *is directed and* $x = \sup \mathrm{approx}(x)$ *(this property is called* algebraicity of D*).*
(iv) If the subset A *of* D *is consistent (there exists* $x \in D$ *such that* $a \sqsubseteq x$ *for all* $a \in A$*), then* $\sup A$ *exists in* D *(this property is called* consistent completeness of D*).*

Several important facts emerge from these conditions, including the existence (indeed construction) of fixed points of continuous functions, and the existence of function spaces (the category of domains is cartesian closed). Moreover, the compact elements provide an abstract notion of computability.

Example 4.3 *(i) The set of all partial functions from N^n into N ordered by graph inclusion is a domain (the proto-typical example) whose compact elements are the finite functions.*
(ii) $(I_P, \subseteq)$ is a domain whose compact elements are the finite subsets of B_P.
(iii) $(I_{P,3}, \subseteq)$ is a domain whose compact elements are the pairs (C_1, C_2) of disjoint finite subsets of B_P.

We now cast an arbitrary domain into an ultrametric space. For this purpose, let γ denote an arbitrary countable ordinal, and let Γ_γ denote the set $\{2^{-\alpha}; \alpha < \gamma\}$ of symbols $2^{-\alpha}$ ordered by $2^{-\alpha} < 2^{-\beta}$ if and only if $\beta < \alpha$.

Definition 4.4 *Let $r : D_C \to \gamma$ be a function, called a* rank function*, form $\Gamma_{\gamma+1}$ and denote $2^{-\gamma}$ by 0. Define $d_r : D \times D \to \Gamma_{\gamma+1}$ by $d_r(x, y) = \inf\{2^{-\alpha}; c \sqsubseteq x$ if and only if $c \sqsubseteq y$ for every $c \in D_C$ with $r(c) < \alpha\}$.*

Then (D, d_r) is an ultrametric space said to be *induced by r*. The definition of d_r is a variation of a construction made by M.B. Smyth in [31, Example 5], and applied to level mappings in logic programming in [28]. Indeed, the intuition behind d_r is that two elements x and y of the domain D are "close" if they dominate the same compact elements up to a certain rank (and hence *agree* in this sense up to this rank); the higher the rank giving agreement, the closer are x and y. Furthermore, (D, d_r) is spherically complete. The proof of this claim does not make use of the existence of a bottom element of D, so this requirement can be omitted. The main idea of the proof is captured in the following lemma, which shows that chains of balls give rise to chains of elements in the domain. We will give the proof in detail since it is a generalization of the original result given in [29] under a directedness condition on the rank function which we now remove. The proof depends on the following elementary facts, see [20].

Fact 4.5 *(1) If $\gamma \leq \delta$ and $x \in B_\delta(y)$, then $B_\gamma(x) \subseteq B_\delta(y)$. Hence every point of a ball is also its centre.*
(2) If $B_\gamma(x) \subset B_\delta(y)$, then $\delta \not\leq \gamma$ (thus $\gamma < \delta$, if Γ is totally ordered).

It will simplify notation in the following proof to denote the ball $B_{2^{-\alpha}}(x)$ by $B^\alpha(x)$.

Lemma 4.6 *Let $B^\beta(y)$ and $B^\alpha(x)$ be arbitrary balls in (D, d_r). Then the following statements hold.*
(1) For any $z \in B^\beta(y)$, we have $\{c \in \mathrm{approx}(z); r(c) < \beta\} = \{c \in \mathrm{approx}(y); r(c) < \beta\}$.
(2) $B_\beta = \sup\{c \in \mathrm{approx}(y); r(c) < \beta\}$ and $B_\alpha = \sup\{c \in \mathrm{approx}(x); r(c) < \alpha\}$ both exist.
(3) $B_\beta \in B^\beta(y)$ and $B_\alpha \in B^\alpha(x)$.
(4) Whenever $B^\alpha(x) \subseteq B^\beta(y)$, we have $B_\beta \sqsubseteq B_\alpha$.

Proof (1) Since $d_r(z, y) \leq 2^{-\beta}$, the first statement follows immediately from the definition of d_r.
(2) Since the set $\{c \in \mathrm{approx}(z); r(c) < \beta\}$ is bounded by z, for any z and β, the second statement follows immediately from the consistent completeness of D.
(3) By definition, we obtain $B_\beta \sqsubseteq y$. Since B_β and y agree on all $c \in D_C$ with $r(c) < \beta$, the first statement in (3) holds, and the second similarly.
(4) First note that $x \in B^\beta(y)$, so that $B^\beta(y) = B^\beta(x)$ and the hypothesis can be written as $B^\alpha(x) \subseteq B^\beta(x)$. We consider two cases.

(i) If $\beta \leq \alpha$, then using (1) and noting again that $x \in B^\beta(y)$ we get $B_\beta = \sup\{c \in \operatorname{approx}(y); r(c) < \beta\} = \sup\{c \in \operatorname{approx}(x); r(c) < \beta\} \sqsubseteq \sup\{c \in \operatorname{approx}(x); r(c) < \alpha\} = B_\alpha$ as required.
(ii) If $\alpha < \beta$, then we cannot have $B^\alpha(x) \subset B^\beta(x)$ and we therefore obtain $B^\alpha(x) = B^\beta(x)$ and consequently $B^\alpha(B_\beta) = B^\beta(B_\beta) = B^\beta(B_\alpha)$ using (3). With the argument of (i) and noting this time that $y \in B^\alpha(x)$, it follows that $B_\alpha \sqsubseteq B_\beta$. We want to show that $B_\alpha = B_\beta$. Assume in fact that $B_\alpha \sqsubset B_\beta$. Since any point of a ball is its centre, we can take $z = B_\beta$ in (1), twice, to obtain $B_\beta = \sup\{c \in \operatorname{approx}(B_\beta); r(c) < \beta\}$ and $B_\alpha = \sup\{c \in \operatorname{approx}(B_\beta); r(c) < \alpha\}$. Thus, the supposition $B_\alpha \sqsubset B_\beta$ means that $\sup\{c \in \operatorname{approx}(B_\beta); r(c) < \alpha\} \sqsubset \sup\{c \in \operatorname{approx}(B_\beta); r(c) < \beta\}$. Since $\{c \in \operatorname{approx}(B_\beta); r(c) < \alpha\} \subseteq \{c \in \operatorname{approx}(B_\beta); r(c) < \beta\}$, there must be some $d \in \{c \in \operatorname{approx}(B_\beta); r(c) < \beta\}$ with $d \not\sqsubseteq \sup\{c \in \operatorname{approx}(B_\beta); r(c) < \alpha\} = B_\alpha$. Thus, there is an element $d \in D_C$ with $r(d) < \beta$ satisfying $d \not\sqsubseteq B_\alpha$ and $d \sqsubseteq B_\beta$. This contradicts the fact that $d_r(B_\alpha, B_\beta) \leq 2^{-\beta}$. Hence, $B_\alpha \not\sqsubset B_\beta$, and since $B_\alpha \sqsubseteq B_\beta$, it follows that $B_\alpha = B_\beta$ and therefore that $B_\beta \sqsubseteq B_\alpha$ as required. □

Given a (decreasing) chain of balls, one now obtains an (increasing) chain of elements in the domain, which has a supremum. This supremum turns out to be contained in the intersection of the chain of balls. Details of the proof can be found in [29]. Thus we obtain the following result.

Theorem 4.7 *The ultrametric space (D, d_r) is spherically complete.*

4.2 Application to Locally Stratified Programs. Suppose P is a normal logic program. Then, as already noted, I_P can be thought of as a domain whose compact elements are the interpretations corresponding to the finite subsets of B_P. Now suppose that P is locally stratified relative to the level mapping $l : B_P \to \gamma$. We define the rank function r_l *induced by* l by setting $r_l(I) = \max\{l(A); A \in I\}$ for every finite $I \neq \emptyset$ and take $r_l(\emptyset) = 0$. Denote the ultrametric resulting from r_l by d_l.

We are now in a position to establish the following result.

Theorem 4.8 *Let P be a normal logic program which is locally stratified with respect to a level mapping l. Then P has a supported model. If, further, P is locally hierarchical with respect to l, then P has a unique supported model.*

Proof According to Theorem 4.7, (I_P, d_l) is spherically complete (this was shown directly for the case of I_P in [21], but follows from our more general result as just stated). Moreover, it was shown in [6] that T_P is contracting since P is locally stratified, and that there cannot exist a ball $B_\pi(J)$ in (I_P, d_l) such that $d(I, T_P(I)) = \pi$ for all $I \in B_\pi(J)$. Therefore, it follows from Theorem 3.5 that T_P has a fixed point and hence that P has a supported model.

Next, if P is locally hierarchical, it was shown in [29] that T_P is strictly contracting. Therefore, by Theorem 3.5 again, it follows that T_P has a unique fixed point and so P has a unique supported model, as required. □

In the same sort of way, the domain $I_{P,3}$ can be turned into an ultrametric space and we obtain a result corresponding to Theorem 4.8. In particular, we see that for locally hierarchical programs P, both Φ_P and the related operator Φ_{P^*}, defined in [14], have a unique fixed point. Programs for which Φ_{P^*} possesses a unique fixed point are interesting and important insomuch as many of the standard models coincide for them, and therefore, for such programs, the various ways of viewing non-monotonic reasoning coincide. The locally hierarchical programs have this property and so, too, do the acceptable programs of [2]. Classes of programs with this property have elsewhere been called *unique supported model classes* by the authors, and characterized in [13, 14, 16] in terms of the fixed points of Φ_{P^*} in various three-valued logics. Theorem 4.8 shows that ultrametric methods and Theorem 3.5 are powerful tools in carrying out investigations of this type.

5 The Stable Model Semantics

When studying non-monotonic reasoning and deductive databases, it is often convenient to consider extended disjunctive logic programs and to allow two different kinds of negation. One of these is interpreted as classical negation and the other is interpreted procedurally as negation as failure, see [19] for this notion. We introduce the following terminology following [11, 17] closely.

Let $\mathcal{L}$ denote a first-order language. By a *literal* L in $\mathcal{L}$ we mean either an atom A or its negation, $\neg A$, where A is an atom in $\mathcal{L}$; a literal is called *ground* if it contains no variable symbols. We denote the set of all ground literals in $\mathcal{L}$ by $\mathtt{Lit}$. A *rule* r in $\mathcal{L}$ is a universally quantifed expression of the following type

$$L_1 \vee \cdots \vee L_n \leftarrow L_{n+1} \wedge \cdots \wedge L_m \wedge \mathtt{not} L_{m+1} \wedge \cdots \wedge \mathtt{not} L_k,$$

where each $L_i \in \mathtt{Lit}$. Given such a rule r, we define $\mathtt{Head}(r) = \{L_1, \ldots, L_n\}$, $\mathtt{Pos}(r) = \{L_{n+1}, \ldots, L_m\}$ and $\mathtt{Neg}(r) = \{L_{m+1}, \ldots, L_k\}$. The keyword $\mathtt{not}$ may be interpreted as negation as failure. An (*extended disjunctive*) *program* Π is a set of (disjunctive) rules. The term "extended" refers to the fact that two kinds of negation are employed, and the term "disjunctive" refers to the appearance of more than a single literal in the heads of rules and to the disjunction between them. A normal logic program can therefore be understood as a special type of extended disjunctive program.

We note that a program is usually defined as a finite set of rules as above, but the literals L_i are allowed to be non-ground. However, we can always replace a program by the set of all ground instances of its rules. This will yield an infinite set if function symbols are present, and a finite set otherwise (in which case Π is called an extended disjunctive database). Either way, in the sequel we assume that all the rules in an extended program are ground. Finally, a rule r, as above, will usually be written in the form

$$L_1, \ldots, L_n \leftarrow L_{n+1}, \ldots, L_m, \mathtt{not} L_{m+1}, \ldots, \mathtt{not} L_k.$$

5.1 Stable Model Semantics. Given a set Π of ground rules as just defined, it is possible to define a multivalued version T_Π of the single-step operator, to define supported models of Π, and to show that these coincide with the fixed points of T_Π, see [12]. Thus, fixed points of multivalued mappings and, consequently, corresponding fixed-point theorems, enter very generally into the discussion. We shall not, however, pursue this line here in complete generality. Instead, we briefly consider another multivalued operator which encapsulates a view of non-monotonic reasoning due to Gelfond and Lifschitz. This leads to the well-known concept of *stable model*, and we show how its existence can be derived from Theorem 3.3.

In order to describe the stable model semantics or answer set semantics for programs, we first consider programs without negation, $\mathtt{not}$. Thus, let Π denote a disjunctive program in which $\mathtt{Neg}(r)$ is empty for each rule $r \in \Pi$. A subset X of $\mathtt{Lit}$, i.e. $X \in 2^{\mathtt{Lit}}$, is said to be *closed by rules in* Π if, for every rule $r \in \Pi$ such that $\mathtt{Pos}(r) \subseteq X$, we have that $\mathtt{Head}(r) \cap X \neq \emptyset$. The set $X \in 2^{\mathtt{Lit}}$ is called an *answer set* for Π if it is closed by rules in Π and satisfies:

1. If X contains complementary literals, then $X = \mathtt{Lit}$.
2. X is minimal, that is, if $A \subseteq X$ and A is closed by rules of Π, then $A = X$.

We denote the set of answer sets of Π by $\alpha(\Pi)$.
Now suppose that Π is a disjunctive program that may contain not. For a set $X \in 2^{\mathtt{Lit}}$, consider the program Π^X defined by

1. If $r \in \Pi$ is such that $\mathtt{Neg}(r) \cap X$ is not empty, then we remove r i.e. $r \notin \Pi^X$.
2. If $r \in \Pi$ is such that $\mathtt{Neg}(r) \cap X$ is empty, then the rule r' belongs to Π^X, where r' is defined by $\mathtt{Head}(r') = \mathtt{Head}(r)$, $\mathtt{Pos}(r') = \mathtt{Pos}(r)$ and $\mathtt{Neg}(r') = \emptyset$.

It is clear that the program Π^X does not contain not and therefore $\alpha(\Pi^X)$ is defined. Following Gelfond and Lifschitz [11], we define the operator $GL : 2^{\mathtt{Lit}} \to 2^{2^{\mathtt{Lit}}}$ by $GL(X) = \alpha(\Pi^X)$. Finally, we say that X is an *answer set* or a *stable model* of Π if $X \in \alpha(\Pi^X)$, that is, if $X \in GL(X)$. In other words, X is an answer set of Π if it is a *fixed point* of the multivalued mapping GL. Again, we use the notation $\alpha(\Pi)$ for the set of answer sets of Π in the general case.

The following example will help to illustrate these ideas.

Example 5.1 *Take* Π *as follows:*

$$p(0) \vee q(0) \leftarrow$$
$$p(a) \vee q(0) \leftarrow q(0) \wedge \neg p(0).$$

If X is any set of literals not containing $p(0)$, then Π^X is the program

$$p(0) \vee q(0) \leftarrow$$
$$p(a) \vee q(0) \leftarrow q(0),$$

and the answer sets of Π^X are $\{p(0)\}$ and $\{q(0)\}$. Thus, $\alpha(\Pi^X) = \{\{p(0)\}, \{q(0)\}\}$. Since $X = \{q(0)\}$ is a suitable choice of X in that it does not contain $p(0)$, we see that $X \in \alpha(\Pi^X)$ and hence that $\{q(0)\}$ is an answer set for Π.

On the other hand, suppose that X is any set of literals which does contain $p(0)$. In this case, the program Π^X is as follows:

$$p(0) \vee q(0) \leftarrow .$$

Again, the only answer sets of Π^X are $\{p(0)\}$ and $\{q(0)\}$. Since $X = \{p(0)\}$ is a suitable choice of X in that it does contain $p(0)$ this time, we see that $\{p(0)\}$ is an answer set for Π, and indeed is the only one other than $\{q(0)\}$. Thus, $\alpha(\Pi) = \{\{p(0)\}, \{q(0)\}\}$.

In this example, $GL(X)$ contains the two elements $\{p(0)\}$ and $\{q(0)\}$ for any set X of literals, and hence is multivalued. Moreover, both $\{p(0)\}$ and $\{q(0)\}$ are fixed points of GL.

5.2 The Generalized Metric of Khamsi, Kreinovich and Misane.

Definition 5.2 *An extended disjunctive program Π is called* locally stratified *if there exists a mapping (a level mapping) $l : \mathtt{Lit} \to \gamma$, where γ is a countable ordinal, such that for every (ground) rule r of Π, where r has the form $L_1, \ldots, L_n \leftarrow L_{n+1}, \ldots, L_m, \mathtt{not}L_{m+1}, \ldots, \mathtt{not}L_k$, the following inequalities hold: $l(L) \geq l(L^{'})$ and $l(L) > l(L^{''})$, where $L, L^{'}$ and $L^{''}$ denote, respectively, elements of $\mathtt{Head}(r), \mathtt{Pos}(r)$ and $\mathtt{Neg}(r)$.*

This definition clearly generalizes Definition 4.1.

In [17], a notion of generalized metric was introduced in order to study the stable model semantics of locally stratified programs. As it turns out, the notion of generalized metric defined in [17] is closely related to the notion of ultrametric introduced in Section 3, and we discuss this connection below.

A *semigroup* is a set V together with an associative binary operation $+ : V \times V \to V$. If $+$ is also commutative, then the semigroup is called *commutative* or *Abelian*. A semigroup is called a *semigroup with* 0 if there exists an element $0 \in V$ such that $0 + u = u$ for all $u \in V$.

By an *ordered semigroup with* 0, we mean a semigroup with 0 on which there is defined an ordering $\leq$ satisfying: $0 \leq v$ for all $v \in V$, and if $v_1 \leq v_2$ and $v_1' \leq v_2'$, then $v_1 + v_1' \leq v_2 + v_2'$.

Definition 5.3 *Let V be an ordered Abelian semigroup with* 0 *and let M be an arbitrary set. A* generalized metric *on M is a mapping $d : M \times M \to V$ which satisfies the following conditions for all $x, y, z \in M$.*
(i) $d(x, y) = 0$ iff $x = y$.
(ii) $d(x, y) = d(y, x)$.
(iii) $d(x, z) \leq d(x, y) + d(y, z)$.
A pair (M, d) consisting of a set M and a generalized metric d on M is called a generalized metric space.

Definition 5.4 *Let V denote the set of all expressions of the type* 0 *or* $2^{-\alpha}$, *where α is a countable ordinal. As before, an order is defined on V by: $0 \leq v$ for every $v \in V$, and $2^{-\alpha} \leq 2^{-\beta}$ iff $\beta \leq \alpha$. As a semigroup operation $u + v$, we will use the maximum* $\max(u, v)$. *It will be convenient to write $\frac{1}{2}2^{-\alpha} = 2^{-(\alpha+1)}$.*

Definition 5.5 *Assume that α is either a countable ordinal or ω_1, the first uncountable ordinal, and that $\mathbf{v} = (v_\beta)_{\beta<\alpha}$ is a decreasing family of elements of V. Let M be a generalized metric space, and let $(x_\beta)_{\beta<\alpha}$ be a family of elements of M.*
(i) (x_β) is said to $\mathbf{v}$-cluster to $x \in M$ *if, for all β, we have $d(x_\beta, x) < v_\beta$ whenever $\beta < \alpha$.*
(ii) (x_β) is said to be $\mathbf{v}$-Cauchy *if, for all β and γ, we have $d(x_\beta, x_\gamma) < v_\beta$ whenever $\beta < \gamma < \alpha$.*
(iii) M is said to be complete *if for every $\mathbf{v}$, every $\mathbf{v}$-Cauchy family $\mathbf{v}$-clusters to some element of M.*
(iv) A set $A \subseteq M$ will be called complete *if for every $\mathbf{v}$, whenever a $\mathbf{v}$-Cauchy family consists of elements of A, it $\mathbf{v}$-clusters to some element of A.*

A mapping $T : M \to 2^M$ is called a $(\frac{1}{2})$-*contraction* if, for every $x \in M$, for every $y \in M$ and for every $a \in T(x)$, there exists $b \in T(y)$ such that $d(a, b) \leq \frac{1}{2}d(x, y)$.

The following theorem was proved in [17].

Theorem 5.6 *Let M be a complete generalized metric space, let T be a multivalued $(\frac{1}{2})$-contraction on M such that $T(x)$ is not empty for some $x \in M$ (thus, T is not identically empty), and suppose that for every $x \in M$ the set $T(x)$ is complete. Then T has a fixed point.*

We present next some new results relating the results just given to the notion of spherical completeness we discussed earlier.

Let (X, d) be a generalized metric space with respect to V as given in Definition 5.4. Then it is easy to see that d is in fact an ultrametric in the sense of Section 3.

Proposition 5.7 *Let X be a complete generalized ultrametric space with respect to V. Then X is spherically complete in the sense of Section 3.*

Proof Let $\mathcal{B} = (B_{v_\beta}(x_\beta))_\beta$ be a decreasing chain of balls in X, and without loss of generality assume that it is strictly decreasing. We have to show that $\bigcap \mathcal{B} \neq \emptyset$. Let $\mathbf{v} = (v_{\beta+1})_\beta$. Since $\mathcal{B}$ is a chain, it is easy to see that (x_β) is $\mathbf{v}$-Cauchy and therefore, by completeness of X, (x_β) $\mathbf{v}$-clusters to some $x \in X$. By definition, this means that $d(x_\beta, x) < v_{\beta+1} < v_\beta$ and therefore that $x \in B_{v_\beta}(x_\beta)$ for all β. Thus, $x \in \bigcap \mathcal{B}$. $\square$

Proposition 5.8 *Let (X, d, V) be a spherically complete ultrametric space in the sense of Section 3. Then X is complete in the sense of the present section.*

Proof Let $\mathbf{v}$= (v_β) be a decreasing family of elements of V which is, without loss of generality, strictly decreasing, and let (x_β) be $\mathbf{v}$-Cauchy. Then $\mathcal{B} = (B_{v_{\beta+1}}(x_\beta))$ is a decreasing chain of balls in X. By spherical completeness, it has non-empty intersection. Choose $x \in \bigcap \mathcal{B}$. Then, for all β we obtain $d(x_\beta, x) \leq v_{\beta+1} < v_\beta$, so that (x_β) $\mathbf{v}$-clusters to x as required. $\square$

This means, by virtue of Theorem 3.3, that we can reformulate the assumptions in Theorem 5.6 and thereby obtain the following theorem. In fact, our conclusion relative to the second statement in this theorem is a special case of Theorem 3.4.

Theorem 5.9 *Let X be a spherically complete ultrametric space (with respect to V) and let f be multivalued, non-empty and strictly contracting on X. Then either of the following conditions ensures the existence of a fixed point of f.*

(i) *The set $\{d(x, y) \mid y \in f(x)\}$ has a minimum in X for all $x \in X$.*
(ii) *The set $f(x)$ is spherically complete for each $x \in X$.*

We close by showing that the existence of a stable model for a locally stratified extended disjunctive logic program Π follows from Proposition 5.7 and Theorem 5.9, and hence, ultimately, from Theorem 3.3. Thus, Theorem 3.3 gives a unified treatment of the fixed-point theory of locally stratified programs and extended disjunctive programs.

Proceeding somewhat along the lines of Definition 4.4, first let $\mathtt{Lit}_\alpha$ denote the set $\{L \in \mathtt{Lit}; l(L) = \alpha\}$, where l is the level mapping with respect to which Π is locally stratified. We now define a generalized metric d on $2^{\mathtt{Lit}}$ as follows: if $A = B$, then $d(A, B) = 0$; if $A \neq B$, then $d(A, B) = 2^{-\alpha}$, where α is the smallest ordinal for which $A \cap \mathtt{Lit}_\alpha \neq B \cap \mathtt{Lit}_\alpha$. The resulting generalized metric space turns out to be complete, as shown in [17]. It is straightforward to see that the GL operator satisfies the assumptions of Theorem 5.9. Therefore, GL has a fixed point which is a stable model of Π.

References

[1] 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.

[2] Apt, K.R. and Pedreschi, D. *Reasoning About Termination of Pure Prolog Programs.* Information and Computation **106** (1) (1993), 109–157.

[3] Apt, K.R., Marek, V.W., Truszczynski, M. and Warren, D.S. *The Logic Programming Paradigm: A 25-Year Perspective*, Springer, Berlin, 1999.

[4] Batarekh, A. and Subrahmanian, V.S. *Topological Model Set Deformations in Logic Programming.* Fundamenta Informaticae **12** (3) (1989), 357–400.

[5] Blair, H.A., Dushin, F., Jakel, D.W., Rivera, A.J. and Sezgin, M. *Continuous Models of Computation for Logic Programs*, in: Apt, K.R., Marek, V.W., Truszczynski, M. and Warren, D.S. (Eds.) *The Logic Programming Paradigm: A 25 Year Perspective*, Springer, Berlin, 1999, pp. 231–255.

[6] Bouamama, S., Misane, D. and Priess-Crampe, S. *An Application of Ultrametric Spaces in Logic Programming.* Preprint, July 1999, pp. 1–5.

[7] Cavedon, L. *Acyclic Logic Programs and the Completeness of SLDNF-Resolution.* Theoretical Computer Science **86** (1991), 81–92.

[8] Fitting, M. *A Kripke-Kleene Semantics for General Logic Programs.* J. Logic Programming **2** (1985), 295–312.

[9] Fitting, M. *Metric Methods: Three Examples and a Theorem.* J. Logic Programming **21** (3) (1994), 113–127.

[10] Van Gelder, A., Ross, K.A. and Schlipf, J.S. *The Well-Founded Semantics for General Logic Programs.* Journal of the ACM **38** (3) (1991), 620–650.

[11] Gelfond, M. and Lifschitz, V. *Classical Negation in Logic Programs and Disjunctive Databases.* New Generation Computing **9** (1991), 365–385.

[12] Hitzler, P. and Seda, A.K. *Multivalued Mappings, Fixed-Point Theorems and Disjunctive Databases*, in: Butterfield, A. and Flynn, S. (Eds.) *Proc. 3rd Irish Workshop on Formal Methods (IWFM'99)*, Electronic Workshops in Computing (eWiC), British Computer Society, 1999, pp. 1–18.

[13] Hitzler, P. and Seda, A.K. *Acceptable Programs Revisited*, in: *Proc. Workshop on Verification in Logic Programming*, 16th Int. Conf. on Logic Programming (ICLP'99), Las Cruces, New Mexico, November 1999, Electronic Notes in Theoretical Computer Science, Volume 30, No 1, Elsevier, pp. 1–18.

[14] Hitzler, P. and Seda, A.K. *Characterizations of Classes of Programs by Three-Valued Operators*, in: Gelfond, M., Leone, N. and Pfeifer, G. (Eds.) *Logic Programming and Nonmonotonic Reasoning*, Proc. 5th Int. Conf. on Logic Programming and Non-Monotonic Reasoning (LP-NMR'99), El Paso, Texas, USA, December 1999. Lecture Notes in Artificial Intelligence, Vol. 1730, Springer, Berlin, 1999, pp. 357–371.

[15] Hitzler, P. and Seda, A.K. *Some Issues Concerning Fixed Points in Computational Logic: Quasi-Metrics, Multivalued Mappings and the Knaster-Tarski Theorem.* Preprint, Department of Mathematics, University College Cork, 1999, pp. 1–15.

[16] Hitzler, P. and Seda, A.K. *A Topological View of Acceptability.* Preprint, Department of Mathematics, University College Cork, 2000, pp. 1–14.

[17] Khamsi, M.A., Kreinovich, V. and Misane, D. *A New Method of Proving the Existence of Answer Sets for Disjunctive Logic Programs: A Metric Fixed-Point Theorem For Multivalued Mappings*, in: Baral, C. and Gelfond, M. (Eds.) *Proc. of the Workshop on Logic Programming with Incomplete Information*, Vancouver, B.C., Canada, October 1993, pp. 58–73.

[18] Kuhlmann, F.V. *A Theorem about Maps on Spherically Complete Ultrametric Spaces, and its Applications.* Preprint, Department of Mathematics and Statistics, University of Saskatchewan in Saskatoon, 1999, pp. 1–20.

[19] Lloyd, J.W. *Foundations of Logic Programming*, 2nd Edition, Springer, Berlin, 1988.

[20] Priess-Crampe, S. and Ribenboim, P. *Fixed Points, Combs and Generalized Power Series.* Abh. Math. Sem. Univ. Hamburg **63** (1993), 227–244.

[21] Priess-Crampe, S. and Ribenboim, P. *Ultrametric Spaces and Logic Programming.* J. Logic Programming **42** (2000), 59–70.

[22] Priess-Crampe, S. and Ribenboim, P. *Logic Programming and Ultrametric Spaces.* Rendiconti di Matematica **Serie VII** (2000), to appear, pp. 1–13.

[23] Priess-Crampe, S. and Ribenboim, P. *Fixed Point and Attractor Theorems for Ultrametric Spaces.* Forum Math. **12** (2000), 53-64.

[24] Przymusinski, T. *On the Declarative Semantics of Deductive Databases and Logic Programs*, in: Minker, J. (Ed.) *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann Publishers Inc., Los Altos, 1988, pp. 193–216.

[25] Ribenboim, P. *The New Theory of Ultrametric Spaces.* Periodica Mathematica Hungarica **32** (1–2) (1996), 103-111.

[26] Rutten, J.J.M.M. *Elements of Generalized Ultrametric Domain Theory.* Theoretical Computer Science **170** (1996), 349–381.

[27] Seda, A.K. *Topology and the Semantics of Logic Programs.* Fundamenta Informaticae **24** (4) (1995), 359–386.

[28] Seda, A.K. *Quasi-metrics and the Semantics of Logic Programs.* Fundamenta Informaticae **29** (1) (1997), 97–117.

[29] Seda, A.K. and Hitzler, P. *Topology and Iterates in Computational Logic.* Proc. of the 12th Summer Conference on Topology and its Applications: Special Session on Topology in Computer Science, Ontario, August 1997, Topology Proceedings **22** (1999), 427–469.

[30] Seda, A.K. and Hitzler, P. *Strictly Level-Decreasing Logic Programs*, in: Butterfield, A. and Flynn, S. (Eds.) *Proc. 2nd Irish Workshop on Formal Methods (IWFM'98)*, Cork, 1998, Electronic Workshops in Computing (eWiC), British Computer Society, 1999, pp. 1–18.

[31] Smyth, M.B. *Totally Bounded Spaces and Compact Ordered Spaces as Domains of Computation*, in: Reed, G.M., Roscoe, A.W. and Wachter, R.F. (Eds.) *Topology and Category Theory in Computer Science*, Oxford University Press, 1991, pp. 207–229.

[32] Stoltenberg-Hansen, V., Lindström, I. and Griffor, E.R. *Mathematical Theory of Domains*, Cambridge Tracts in Theoretical Computer Science No. 22, Cambridge University Press, Cambridge, 1994.