METRIC METHODS
THREE EXAMPLES AND A THEOREM

MELVIN FITTING

* †

▷ The existence of a model for a logic program is generally established by lattice-theoretic arguments. We present three examples to show that metric methods can often be used instead, generally in a direct, straightforward way. One example is a game program, which is not stratified or locally stratified, but which has a unique supported model whose existence is easily established using metric methods. The second example is a program without a unique supported model, but having a part that is 'well-behaved.' The third example is a program in which one part depends on another, illustrating how modularity might be treated metrically. Finally we use ideas from this third example to prove a general result from [3]. The intention in presenting these examples and the theorem is to stimulate interest in metric techniques, and is not to present a fully developed theory. ◁

1. INTRODUCTION

One of the most common approaches to the problem of assigning a meaning to a logic program is to associate a single-step function with the program, and then look for a fixed point. If the underlying logic is classical, the operator associated with program $\mathcal{P}$ is denoted $T_{\mathcal{P}}$. The existence of a fixed point for $T_{\mathcal{P}}$ usually follows from lattice-theoretic considerations. For example, if $\mathcal{P}$ has no negations $T_{\mathcal{P}}$ will be monotonic on the lattice of valuations, and the Knaster-Tarski theorem applies. If $\mathcal{P}$ has negations $T_{\mathcal{P}}$ will no longer be monotonic in this lattice, and so the existence

*Address correspondence to*Department of Computer Science, Graduate Center (CUNY), 33 West 42nd Street, New York, NY 10036

*Research partly supported by NSF Grant CCR-9104015.

†I want to thank one of the referees, whose suggestions improved the presentation in the paper.

655 Avenue of the Americas, New York, NY 10010 0743-1066/93/$3.50

of a fixed point can no longer be guaranteed. To deal with this a three-valued approach was developed in [10] and elsewhere. Instead of $T_{\mathcal{P}}$, which maps classical valuations to classical valuations, an operator $\Phi_{\mathcal{P}}$ is considered which maps three-valued valuations to three-valued valuations. Using an ordering corresponding to 'degree of information' instead of 'degree of truth,' $\Phi_{\mathcal{P}}$ will always be monotonic and so the existence of a fixed point is again guaranteed, by a generalization of the Knaster-Tarski theorem.

Sometimes the three-valued approach is natural, but there are examples where it is quite awkward. For instance, there is a simple game-playing program from [11] that causes considerable difficulties. Suppose there is a game, $\mathcal{G}$, with positions denoted by constants, a, b, ... (If the game is chess, for instance, think of a constant as coding up the location of all the pieces, together with the information on whose move it is, plus other information such as how many times a board arrangement has recurred.) By convention a loss for a player of $\mathcal{G}$ means the player has no move. Now, the following program $\mathcal{P}(\mathcal{G})$ captures the notion of winning in $\mathcal{G}$:

$$\texttt{win}(X) \leftarrow \texttt{move}(X, Y), \neg\texttt{win}(Y)$$

supplemented with a list of facts, $\texttt{move}(a, b)$, $\texttt{move}(c, d)$, ..., listing all the legal moves of game $\mathcal{G}$. The intention is, $\leftarrow \texttt{win}(c)$ should succeed if and only if there is a possible win starting from position c.

There is essentially only one meaning that can be attached to this program. In fact, the operator $T_{\mathcal{P}(\mathcal{G})}$ has a unique (two-valued) fixed point — the problem comes in showing that this is so since $T_{\mathcal{P}(\mathcal{G})}$ is not monotone. The program $\mathcal{P}(\mathcal{G})$ is not stratified; it is not even locally stratified. It is remarked in [14] that this program "... is one of the examples that led to the formulation of well-founded semantics, as well as stable models." In [3] it is shown that several semantical approaches agree for this program, but the argument is quite roundabout. First the existence of a three-valued model is shown, using the operator $\Phi_{\mathcal{P}(\mathcal{G})}$, then by an induction argument it is shown that the model is actually two-valued.

Much of the difficulty arises from the adherence to lattice-theoretic techniques. Alternative approaches based on metric methods have appeared from time to time, but for various reasons they have never become well-known in the logic programming community, though they are fairly common elsewhere. The purpose of this paper is to demonstrate their utility and simplicity (when they are applicable). We do this by presenting three examples of programs, and a general theorem, all of which are subject to metric methods. One of these is a direct argument that the game program operator $T_{\mathcal{P}(\mathcal{G})}$ has a unique fixed point and that it coincides with the unique fixed point of $\Phi_{\mathcal{P}(\mathcal{G})}$. The argument is simpler and more direct than that based on other techniques.

The basic theory underlying a metric approach was developed in considerable detail in [4], which was not published. Additional material can be found in [5]. It is not our intention here to be complete and exhaustive. Instead we want to demonstrate utility, in order to encourage others to try applying these and related techniques. As far as possible we try to keep this paper self-contained.

2. VALUATIONS AND METRICS

Logic programming semantics is generally based on classical, two-valued logic. It has been found useful sometimes to base it on partial, or three-valued logic. Less frequently, bilattices or other more esoteric spaces of truth values are used. What we are about to say does not depend critically on the exact choice. We carry out most of the work using ordinary two-valued logic and mention three-valued briefly from time to time. We assume some program $\mathcal{P}$ has been specified, and all references to ground atoms involve the Herbrand base of $\mathcal{P}$.

Definition2.1. A *valuation* is a mapping from ground atoms to $\{false, true\}$. It is extended to literals by mapping $\neg A$ to *false* (*true*) if A maps to *true* (*false*).

What we must do is turn the collection of valuations into a metric space. We begin by recalling the definition.

Definition2.2. A *metric* or distance function on a space $\mathcal{M}$ is a mapping

$$d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$$

(where $\mathbb{R}$ is the real numbers) such that:

1. $d(x, y) = 0$ if and only if $x = y$,
2. $d(x, y) = d(y, x)$,
3. $d(x, y) \leq d(x, z) + d(z, y)$.

Occasionally we will also need the notion of a *pseudo-metric*. This is like a metric, except that one drops the condition: $d(x, y) = 0$ implies $x = y$.

There is a simple and quite general way of introducing a metric on the set of valuations, involving the notion of level mapping, [8, 9, 6, 7, 2].

Definition2.3. A *level mapping* for a program $\mathcal{P}$ is a function $| \ | : B_P \to \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers and B_P is the Herbrand base for $\mathcal{P}$. If $|A| = n$ we will say the level of A is n. Now, if we are given a level mapping, we can introduce a (candidate for) a metric on the set of valuations.

Definition2.4. Let $| \ |$ be a level mapping. An associated mapping d is defined as follows. Let v and w be two valuations. If $v = w$, set $d(v, w) = 0$. Otherwise, set $d(v, w) = 1/2^n$ where v and w differ on some ground atom A of level n, but agree on all ground atoms of lower level.

The notion of distance defined above is fairly standard. See the last Chapter of [12], for instance, for a very similar notion. All conditions for being a metric space are trivial with the exception of condition 3, the triangle inequality, which we now check.

If any two of x, y, or z are the same, condition 3 is easily verified. Now say x, y, and z are valuations with $d(x, y) = 1/2^n$, $d(x, z) = 1/2^m$, and $d(z, y) = 1/2^k$. Then x and z differ on a ground atom of level m, but agree on all ground atoms with

level $m-1$ or less. Similarly z and y agree on all ground atoms with level $k-1$ or less. Say $m \leq k$ (the argument is similar the other way around). Then x and y must agree on all ground atoms with level $m-1$ or less and so $d(x,y) \leq 1/2^m$. Consequently $d(x,y) \leq 1/2^m + 1/2^k = d(x,z) + d(z,y)$.

The proof just given actually shows the stronger fact that d is an *ultrametric*, that is, $d(x,y) \leq \max\{d(x,z), d(z,y)\}$, though we do not make use of this here. Further, the notion of distance defined above not only makes the space of valuations into a metric (or ultrametric) space, it actually gives us a *complete* metric space. There are several equivalent ways completeness can be characterized; we give one standard version for reference.

Definition2.5. A metric space $\mathcal{M}$ is *complete* if every Cauchy sequence converges. A sequence $s_1, s_2, s_3, \ldots$ is *Cauchy* if, for every $\epsilon > 0$ there is an integer N such that for all $n, m \geq N$, $d(s_n, s_m) \leq \epsilon$. (Essentially a sequence is Cauchy if its elements get arbitrarily close together.) The sequence *converges* if there is an s such that, for every $\epsilon > 0$, there is an integer N so that for all $n \geq N$, $d(s_n, s) \leq \epsilon$. (Essentially a sequence converges if there is some value its elements get arbitrarily close to.)

Proposition 1. The space of valuations, using a metric based on a level mapping, is a complete metric space.

The proof of completeness is quite simple. We sketch the ideas. Suppose s_1, s_2, s_3, ... is a Cauchy sequence of valuations. We define a 'limit' valuation s as follows. Suppose A is a ground atom of level n; we say what value s should assign to A. Since the sequence is Cauchy, from some point on all valuations are within $1/2^{n+1}$ of each other, and so agree on all ground atoms of level $\leq n$. Then from some point on all valuations in the sequence assign the same truth value to A. Set $s(A)$ to be this value. In this way a valuation s is characterized, and it is straightforward to check that it is actually the limit of the sequence.

Finally, we were working above with classical valuations. A little checking will verify that everything we said applies with no essential changes to three-valued valuations (or to valuations on several more general spaces of truth values as well). We leave this to you.

3. GENERAL PROGRAMS AND SINGLE-STEP OPERATORS

As usual, a *general program* is made up of a finite number of *general clauses*, which are of the form

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

where A is atomic and $L_1, \ldots, L_n$ are literals. For convenience, we repeat the standard definition of the (two-valued) single-step operator for a general program.

Definition3.1. Let $\mathcal{P}$ be a general program. The associated mapping $T_{\mathcal{P}}$, mapping valuations to valuations, is characterized as follows. For a valuation v and a ground atom A,

1. If there is a ground instance, $A \leftarrow L_1, \ldots, L_n$, of a clause in $\mathcal{P}$ with $v(L_1) =$ *true* and $\ldots$ and $v(L_n) =$ *true*, then $T_{\mathcal{P}}(v)(A) =$ *true*.
2. Otherwise, $T_{\mathcal{P}}(v)(A) =$ *false*. That is, if A is not the head of any ground instance of a clause in $\mathcal{P}$, or if for every ground instance, $A \leftarrow L_1, \ldots, L_n$, of a clause in $\mathcal{P}$, $v(L_1) =$ *false* or $\ldots$ or $v(L_n) =$ *false*, then $T_{\mathcal{P}}(v)(A) =$ *false*.

The machinery of metric spaces was introduced in the previous section. The set of valuations has many metrics, since each level mapping determines one. The central idea is: find a metric with respect to which the single-step operator $T_{\mathcal{P}}$ is a *contraction*. We begin by recalling the definition of a contraction (see [15]), then we say why we are interested in them.

Definition3.2. Let $\mathcal{M}$ be a metric space. A mapping $f : \mathcal{M} \to \mathcal{M}$ is a *contraction* if there is some number $0 < k < 1$ such that for all $x, y \in \mathcal{M}$:

$$d(f(x), f(y)) \leq k \cdot d(x, y).$$

(Informally, a contraction shrinks distances by at least some constant factor that is smaller than 1, independently of the choice of points.)

Our interest in contractions comes from the following, which can sometimes serve as a replacement for the Knaster-Tarski Theorem.

The Banach Contraction Theorem A contraction mapping f on a complete metric space has a unique fixed point. Further, the sequence $x, f(x), f(f(x)), \ldots$ converges to this fixed point for any x.

We do not prove this here. It is standard, and a proof can be found in [15] among other places. But with it available many of our problems are at an end. If $T_{\mathcal{P}}$ is a contraction on the space of valuations, it has a unique fixed point. Further, using essentially the same argument on the metric space of three-valued valuations we can often show the $\Phi_{\mathcal{P}}$ operator is a contraction as well. If so, it also has a unique fixed point. Since every fixed point of $T_{\mathcal{P}}$ is also a fixed point of $\Phi_{\mathcal{P}}$, classical and three-valued semantics would then coincide for $\mathcal{P}$. Since every stable model is a fixed point of $\Phi_{\mathcal{P}}$, there is a unique stable (and well-founded) model. Thus the whole idea reduces to this: *find a metric on the set of valuations that makes $T_{\mathcal{P}}$ and, if possible, $\Phi_{\mathcal{P}}$ into a contraction.*

One more point. The three-valued single-step operator $\Phi_{\mathcal{P}}$ is monotonic in the ordering used in [10], so the usual lattice-theoretic arguments show $\Phi_{\mathcal{P}}$ converges to a fixpoint when started with the valuation mapping all ground atoms to $\bot$. But in general the $\Phi_{\mathcal{P}}$ operator may need more than ω steps to reach its fixpoint. If the approach outlined above can be managed, that can not happen, since the Banach Theorem gives convergence in ω steps, starting from anywhere.

4. THE GAME PROGRAM

We recall the program from Section 1. We are given a game $\mathcal{G}$, with positions denoted by constants. The program $\mathcal{P}(\mathcal{G})$ is the following:

(w) $\texttt{win}(X) \leftarrow \texttt{move}(X,Y), \neg\texttt{win}(Y).$
(m) $\texttt{move}(a_i, a_j) \leftarrow .$ for all legal moves a_i to a_j

This program has two predicates, `move` and `win`, and of the two, `move` is not problematic. All clauses for `move` are simply facts, so if we start with initial valuations that differ, after one cycle of applying either the T or the Φ operator we reach valuations that agree on all instances of `move`. Consequently it will do no harm, and will simplify the presentation, if we ignore the behavior of valuations on `move`. In effect we confine our attention to the behavior of valuations on `win`.

Valuation Assumption *For the rest of this section the notion of valuation for $\mathcal{P}(\mathcal{G})$ is restricted to those that map to true exactly those instances of* `move` *that occur as facts in the program $\mathcal{P}(\mathcal{G})$.*

We need to make certain reasonable assumptions about the game $\mathcal{G}$ if the behavior of the program $\mathcal{P}(\mathcal{G})$ is to be decent. In order to present this simply, we use the following more-or-less standard terminology.

Definition4.1. A *game tree* for a position p is a tree having each node labeled with a position in game $\mathcal{G}$, with p at the root, and with the children of a node labeled with the positions reachable in one move from the position labeling the parent. The *height* of a game tree is the length of the longest branch.

Game Assumption *For the rest of this section we assume that* `move` *is acyclic, and hence every game tree for a position in $\mathcal{G}$ is finite.*

The Game Assumption covers games with no loops, such as chess which terminates if the same position recurs three times, or for 50 moves no pawn has been moved and no piece captured. Indeed, it is an assumption that applies to most 'reasonable' games. We make use of it to define a complete metric on the space of valuations, via a level mapping.

Definition4.2. Let $|\ |$ be the level mapping: $|\texttt{win}(p)|$ is the height of the game tree with p at the root. The metric d is the one induced by this level mapping.

Now, the main item we need.

Proposition 1. The operator $T_{\mathcal{P}(\mathcal{G})}$ is a contraction, using the metric defined above.

The proof of this is simple — we sketch the idea. If the valuations x and y agree on all game trees of height $< n$, $T_{\mathcal{P}(\mathcal{G})}(x)$ and $T_{\mathcal{P}(\mathcal{G})}(y)$ will agree on all game trees of height $< n+1$ since an application of $T_{\mathcal{P}(\mathcal{G})}$ corresponds to the carrying out of one more move in the game. Consequently if $d(x,y) = 1/2^n$ then $d(T_{\mathcal{P}(\mathcal{G})}(x), T_{\mathcal{P}(\mathcal{G})}(y)) = 1/2^{n+1} = (1/2)\cdot(1/2^n) = (1/2)\cdot d(x,y)$.

This essentially ends the discussion of the game program. There is a unique fixed point for $T_{\mathcal{P}(\mathcal{G})}$. And by exactly the same argument, but applied in the three-valued setting, there is a unique fixed point for $\Phi_{\mathcal{P}(\mathcal{G})}$ as well, so classical, three-valued, and stable semantics coincide.

5. A SECOND EXAMPLE

The game program has a unique supported model (supported models are fixed points of the $T_{\mathcal{P}}$ operator). Many programs of interest are not like this but even so, parts of programs may be well-behaved. In this section we look at a contrived example intended to show how such programs can sometimes be treated with metric methods. The program is one for the even numbers, combined with one designed to have as many models as possible. We denote the program $\mathcal{P}(\mathcal{E})$.

(e_1) $\texttt{even}(0) \leftarrow$.
(e_2) $\texttt{even}(s(X)) \leftarrow \neg\texttt{even}(X)$.
(a) $\texttt{any}(X) \leftarrow \texttt{any}(X)$.

The subprogram consisting of (e_1) and (e_2) has a unique supported model, while the subprogram consisting of (a) has infinitely many models. So, essentially, the idea is to ignore the problematic part by factoring it out.

Definition5.1. A *partial level mapping* for a program $\mathcal{P}$ is a function $|\ |: B_0 \rightarrow \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers and B_0 is some subset of the Herbrand base for $\mathcal{P}$. We say A is of level n if A is in the domain of $|\ |$ and $|A| = n$. Now

we proceed rather like before.

Definition5.2. Let $|\ |$ be a partial level mapping. An associated mapping d is defined as follows. Let v and w be two valuations. If v and w agree on the domain of $|\ |$, set $d(v, w) = 0$. Otherwise, set $d(v, w) = 1/2^n$ where v and w differ on some ground atom A of level n, but agree on all ground atoms of lower level.

A mapping deriving from a partial level mapping that is not total will be a pseudo-metric, but not a metric. The triangle inequality is verified exactly as before. The problem, of course, is that v and w might agree on all members of the domain of $|\ |$ but differ elsewhere, and so $d(v, w) = 0$ while $v \neq w$. However, a true metric can be introduced in a relatively painless manner.

Call two members, x and y of a pseudo-metric space *equivalent* if $d(x, y) = 0$. This is easily seen to be an equivalence relation. Partition the pseudo-metric space into equivalence classes; denote the class containing x by $\overline{x}$. Define a mapping $\overline{d}$ on equivalence classes by: $\overline{d}(\overline{x}, \overline{y}) = d(x, y)$. This is well-defined, and is a true metric on the space of equivalence classes.

One still has a notion of *completeness* for a pseudo-metric space. The definition is word for word the same, but now the limit of a Cauchy sequence need not be unique. Instead, any two limits must be a distance of 0 apart. A partial level mapping yields a complete pseudo-metric, using the same argument as before. And when we pass to the metric space of equivalence classes, a complete metric space results.

Now, define a partial level mapping for the program $\mathcal{P}(\mathcal{E})$, with domain the set of ground atoms of the form $\texttt{even}(t)$. Since a ground term for this program must be of the form $s^n(0)$, we simply set $|\texttt{even}(s^n(0))| = n$. This produces a complete pseudo-metric space. Form the space of equivalence classes, as described above, obtaining a complete metric space. The single-step operator $T_{\mathcal{P}(\mathcal{E})}$ induces an operator on the space of equivalence classes:

$$\overline{T_{\mathcal{P}(\mathcal{E})}}(\overline{v}) = \overline{T_{\mathcal{P}(\mathcal{E})}(v)}$$

This is well defined, and is easily verified to be a contraction. Then the operator $\overline{T_{\mathcal{P}(\mathcal{E})}}$ has a unique fixed point. In effect, this means that the behavior of `even` in $\mathcal{P}(\mathcal{E})$ is completely determined, even though that of `any` is not.

6. A FINAL EXAMPLE

We conclude with a program from [3] intended to compute transitive closures, deriving from a program in [13] determining graph connectivity. It is like the game program in having a unique supported model, but also it is like the even number program in having two relations to be taken into account. The complication this example introduces is that one of the relations depends on the other. In effect the example shows how part of a program can be semantically understood, then that understanding used in developing the semantics of the rest.

We give the program, denoted $\mathcal{P}(\mathcal{T})$, using the notation and terminology of [3]. Suppose we have a finite graph with nodes labeled with constants, a, b,..., where $\mathcal{A}$ is the set of labels used.

(r_1) $\mathtt{r}(X, Z, E, G) \leftarrow \mathtt{member}([X, Z], E).$
(r_2) $\mathtt{r}(X, Z, E, G) \leftarrow \mathtt{member}([X, Y], E), \neg\mathtt{member}(Y, G), \mathtt{r}(Y, Z, E, [Y|G]).$
(m_1) $\mathtt{member}(X, [X|T]) \leftarrow .$
(m_2) $\mathtt{member}(X, [Y|T]) \leftarrow \mathtt{member}(X, T).$
(e) $\mathtt{element}(a) \leftarrow .$ for $a \in \mathcal{A}$

An edge of a graph from node a to node b is represented by the two-element list $[a, b]$. Now, if e is a list of all edges for a graph, it is intended that $\leftarrow \mathtt{r}(x, y, e, [])$ should succeed just when there is a path in the graph from x to y; that is, when $[x, y]$ is in the transitive closure of e. In operational terms, the fourth argument of `r` keeps track of which nodes have been visited, so the path search does not become cyclic. Declaratively, as K. Apt has forcefully pointed out, $\mathtt{r}(x, y, e, a)$ should be read: there is a path from x to y in graph e that avoids the nodes in a. The purpose of item (e) is just to make sure all the node labels are in the Herbrand base. In what follows we simply ignore `element`, just as we ignored `move` in the game program.

The graph program is more complicated than the previous example because there are two relations we are concerned with, not one, and they are not independent. In order to deal with this dependency we divide the specification of a metric into two parts, one for each of the two relations. Note that while `r` refers to `member`, the clauses for `member` are self-contained. Consequently we can discuss the behavior of `member` first, without reference to `r`. Once its behavior is determined, we can make use of it in our consideration of `r`, which is rather what one might expect us to do.

There is, however, a minor annoyance concerning `member` that must be faced. There is no type checking to ensure Y is a list in $\mathtt{member}(X, Y)$, and in fact the program will report that $\mathtt{member}(a, [a|b])$ holds, where a and b are constants, even though $[a|b]$ is not a list. Type checking could be added, but doing so complicates the program, and does not change behavior in the cases we are interested in. We have decided to complicate the semantics somewhat instead.

Definition6.1. By a *pseudo-list* we mean a ground term of one of the forms [] and $[a \mid b]$, where a and b are arbitrary terms. We define a notion of *length* as follows. If b is not a pseudo-list, its length is 0. The length of [] is also 0. And the length of $[a \mid b]$ is 1+ the length of b. We also define the notion of *element* as follows. If b is not a pseudo-list, or is [], it has no elements. The elements of $[a \mid b]$ are a, and the elements of b. Note that a list is also a pseudo-list, and for it length and element has the usual meaning. Now we can define a partial level mapping for `member`.

Definition6.2. If y is a pseudo-list, set $|\texttt{member}(x, y)|$ to be the length of y. If y is not a pseudo-list, set $|\texttt{member}(x, y)|$ to be 0. The function $|\ |$ is not defined on ground atoms of the form $\texttt{r}(x, y, e, g)$. The distance function d_1 is the pseudo-metric induced by this partial level mapping.

We can now proceed as in the previous section, forming a metric space of equivalence classes. In this way we can show that the program semantically determines the behavior of `member` uniquely: $\texttt{member}(x, y)$ holds if and only if y is a pseudo-list and x is one of its elements. The details are straightforward, and we omit them.

Once the behavior of `member` has been determined, this behavior can be used to help determine the behavior of `r`; that is, a kind of *modularity* can be invoked. The way this shows up in what follows is quite direct. Where `member` is concerned, we measure distance in terms of how much deviation there is from the real membership in pseudo-lists relation.

Definition6.3. Let v_M be a valuation such that $v_M(\texttt{member}(x, y)) = \mathit{true}$ if and only if y is a pseudo-list and x is one of its elements. On ground atoms of the form $\texttt{r}(x, y, e, g)$, v_M is arbitrary. Now, for valuations v and w, define a distance function d_2 by:

$$d_2(v, w) = \max\{d_1(v, v_M), d_1(w, v_M)\}$$

The function d_2 is also a pseudo-metric. The triangle inequality is verified as follows.

$$\begin{aligned} d_2(x, y) &= \max\{d_1(x, v_M), d_1(y, v_M)\} \\ &\leq \max\{d_1(x, v_M), d_1(z, v_M), d_1(y, v_M)\} \\ &\leq \max\{d_1(x, v_M), d_1(z, v_M)\} + \max\{d_1(z, v_M), d_1(y, v_M)\} \\ &= d_2(x, z) + d_2(z, y) \end{aligned}$$

Now we turn to the behavior of `r`. For this purpose we introduce yet another pseudo-metric d_3, based on a partial level mapping taken directly from [3].

Definition6.4. Let t_1 and t_2 be ground terms; $(t_1 \smallsetminus t_2)$ is the number of ground atoms y such that $[x, y]$ is an element of t_1 (for some x) but y is not an element of t_2. (Recall, terms that are pseudo-lists have elements; terms that are not, do not. The expression $(t_1 \smallsetminus t_2)$ has meaning either way.)

Let $\mathtt{r}(x, z, e, g)$ be ground. By $\|\mathtt{r}(x, z, e, g)\|$ we mean the number: length of e + length of $g+2\cdot(e \smallsetminus g)+1$. (Recall, if either term e or g is not a pseudo-list, its length is 0.) The function $\|\ \|$ is not defined on ground atoms of the form $\mathtt{member}(x, y)$.

The pseudo-metric d_3 is the one induced by the partial level mapping $\|\ \|$.

Finally, we define a metric that combines the two pseudo-metrics d_2 and d_3.

Definition6.5. For valuations v and w, set $d(v, w) = \max\{d_2(v, w), d_3(v, w)\}$.

Whenever a distance function is defined in this way, as the maximum of other pseudo-metrics, the result is always pseudo-metric — this is straightforward to show. Also, if $d(v, w) = 0$ then $d_1(v, v_M) = d_1(w, v_M) = d_3(v, w) = 0$ and it follows that v and w must agree on all instances of both `member` and `r`, and so $v = w$. Thus d is a *metric*. It can also be verified that it gives us a *complete* metric space; we omit details.

Proposition 1. $T_{\mathcal{P}(\mathcal{T})}$ is a contraction.

Proof. If $d(v, w) = 0$ then $v = w$ since d is a metric. Then trivially $T_{\mathcal{P}(\mathcal{T})}(v) = T_{\mathcal{P}(\mathcal{T})}(w)$ and so $d(T_{\mathcal{P}(\mathcal{T})}(v), T_{\mathcal{P}(\mathcal{T})}(w)) = 0$. Now suppose v and w are two valuations with $d(v, w) = 1/2^n$.

First, $d_2(v, w) \leq 1/2^n$, so $d_1(v, v_M) \leq 1/2^n$ and v and v_M agree on all ground atoms of the form $\mathtt{member}(a, l)$ with $|\mathtt{member}(a, l)| < n$, and so agree on ground atoms $\mathtt{member}(a, l)$ where l is a pseudo-list with length $< n$. It follows from (m_1) and (m_2) that $T_{\mathcal{P}(\mathcal{T})}(v)$ and $T_{\mathcal{P}(\mathcal{T})}(v_M)$ agree on ground atoms involving pseudo-lists of length $\leq n$, and so $d_1(T_{\mathcal{P}(\mathcal{T})}(v), T_{\mathcal{P}(\mathcal{T})}(v_M)) \leq 1/2^{n+1}$. But $d_1(T_{\mathcal{P}(\mathcal{T})}(v_M), v_M) = 0$ so by the triangle inequality, $d_1(T_{\mathcal{P}(\mathcal{T})}(v), v_M) \leq 1/2^{n+1}$, and hence

$$d_1(T_{\mathcal{P}(\mathcal{T})}(v), v_M) \leq \frac{1}{2} \cdot d(v, w).$$

Likewise

$$d_1(T_{\mathcal{P}(\mathcal{T})}(w), v_M) \leq \frac{1}{2} \cdot d(v, w).$$

Consequently

$$d_2(T_{\mathcal{P}(\mathcal{T})}(v), T_{\mathcal{P}(\mathcal{T})}(w)) \leq \frac{1}{2} \cdot d(v, w).$$

Now let $\mathtt{r}(x, z, e, g)$ be a ground atom with $\|\mathtt{r}(x, z, e, g)\| \leq n$; we show $T_{\mathcal{P}(\mathcal{T})}(v)$ and $T_{\mathcal{P}(\mathcal{T})}(w)$ must agree on this ground atom. If we do this, it will follow that $d_3(T_{\mathcal{P}(\mathcal{T})}(v), T_{\mathcal{P}(\mathcal{T})}(w)) \leq 1/2^{n+1}$, and so

$$d_3(T_{\mathcal{P}(\mathcal{T})}(v), T_{\mathcal{P}(\mathcal{T})}(w)) \leq \frac{1}{2} \cdot d(v, w)$$

Then we will immediately have

$$d(T_{\mathcal{P}(\mathcal{T})}(v), T_{\mathcal{P}(\mathcal{T})}(w)) \leq \frac{1}{2} \cdot d(v, w)$$

which says that d is a contraction, and we will be finished.

So to complete things, assume $T_{\mathcal{P}(\mathcal{T})}(v)$ assigns $\mathtt{r}(x,z,e,g)$ *true*; we will show $T_{\mathcal{P}(\mathcal{T})}(w)$ also assigns it *true*. Our underlying assumptions are that $\|\mathtt{r}(x,z,e,g)\| \leq n$, and $d(v,w) = 1/2^n$.

Since $d(v,w) = 1/2^n$, $d_1(v,v_M)$ and $d_1(w,v_M)$ are both $\leq 1/2^n$. This implies that v, w, v_M all agree on ground atoms $\mathtt{member}(a,l)$ whenever l is a pseudo-list of length $< n$, and so if l is a pseudo-list of length $< n$, $v(\mathtt{member}(a,l)) = \mathit{true}$ iff $w(\mathtt{member}(a,l)) = \mathit{true}$ iff a is an element of l.

Now, $T_{\mathcal{P}(\mathcal{T})}(v)$ assigns $\mathtt{r}(x,z,e,v)$ the value *true*. There are two cases, depending on whether (r_1) or (r_2) was involved. The case where (r_1) was used is the easier of the two, and is omitted. Suppose (r_2) is the reason that $T_{\mathcal{P}(\mathcal{T})}(v)$ assigns *true* to $\mathtt{r}(x,z,e,g)$. Then there is a ground instance of (r_2):

$$\mathtt{r}(x,z,e,g) \leftarrow \mathtt{member}([x,y],e), \neg\mathtt{member}(y,g), \mathtt{r}(y,z,e,[y|g])$$

and v assigns *true* to $\mathtt{member}([x,y],e)$, $\neg\mathtt{member}(y,v)$, and $\mathtt{r}(y,z,e,[y|g])$. Now, $n \geq \|\mathtt{r}(x,z,e,g)\| >$ length of e, and $v(\mathtt{member}([x,y],e)) = \mathit{true}$, so it really is the case that $[x,y]$ is an element of e, and also $w(\mathtt{member}([x,y],e) = \mathit{true}$. Also $n \geq \|\mathtt{r}(x,z,e,g)\| >$ length of g, and since $v(\neg\mathtt{member}(y,g)) = \mathit{true}$, y is not an element of g and $w(\neg\mathtt{member}(y,g)) = \mathit{true}$.

Finally, $[x,y]$ is an element of e and y is not an element of g so $(e \smallsetminus [y|g]) = (e \smallsetminus g) - 1$. Then

$$\begin{array}{rcl} \|\mathtt{r}(y,z,e,[y|g])\| & = & \text{length of } e + \text{length of } [y|g] + 2\cdot(e \smallsetminus [y|g]) + 1 \\ & = & \text{length of } e + \text{length of } g + 1 + 2\cdot((e \smallsetminus g) - 1) + 1 \\ & = & \text{length of } e + \text{length of } g + 2\cdot(e \smallsetminus g) \\ & < & \|\mathtt{r}(y,z,e,g)\| \\ & \leq & n \end{array}$$

Since $d_3(v,w) \leq 1/2^n$, v and w agree on $\mathtt{r}(y,z,e,[y|g])$, and so $w(\mathtt{r}(y,z,e,[y|g])) = \mathit{true}$. We have seen that $w(\mathtt{member}([x,y],e)) = \mathit{true}$ and $w(\neg\mathtt{member}(y,g)) = \mathit{true}$. Then, using (r_2), $T_{\mathcal{P}(\mathcal{T})}(w)$ must assign $\mathtt{r}(x,z,e,g)$ *true*, and we are done. □

7. ACCEPTABLE PROGRAMS

In [3] the game program $\mathcal{P}(\mathcal{G})$ was shown to be well-behaved semantically by showing it belonged to a class of well-behaved programs, the *acceptable* ones. It was a central result of that paper that for acceptable programs various ways of defining semantics coincide. The method of proof was somewhat roundabout: first the existence of a three-valued model was established, then an induction argument showed it was actually two-valued. In this section we use metric methods to re-prove this result about acceptable programs more directly.

We begin with the necessary background concerning acceptable programs. The underlying idea is to capture in a simple way the left-first approach of Prolog, as contrasted with the non-deterministic selection rule generally considered in theoretical approaches to logic programming The notion of acceptability originated in [2] and was extended to allow negation in [3]. In order to define the class of acceptable programs, we need some preliminary concepts, the primary one being that of a level mapping, which we have already seen. What is added here is the singling out of the essentially negative part of a program. This is taken directly from [3].

Definition7.1. Let P be a program, and p and q be relations.

1. *p refers to q* if there is a clause in P with p in its head and q in its body.
2. *p depends on q* if $p = q$, or there is a sequence $p = p_1, p_2, \ldots, p_n = q$, where p_i refers to p_{i+1}.
3. Neg_P is the set of relations in P which occur in a negative literal in the body of a clause of P.
4. Neg_P^* is the set of relations in P on which the relations in Neg_P depend.
5. P^- is the set of clauses in P whose head contains a relation from Neg_P^*.

Thus the clauses in P^- are those one needs in order to understand the behavior of relations that appear negated in P. *To keep terminology down, if a ground literal L has its relation symbol in Neg_P^*, we will say L itself is in Neg_P^*.* Now for the central notion, presented in a form somewhat modified from [3].

Definition7.2. Let P be a general program, $|\ |$ be a level mapping for P, and I be a model (not necessarily Herbrand) of P whose restrictions to the relations in Neg_P^* is a model for $comp(P^-)$ (where $comp(Q)$ denotes the Clark completion of program Q). P is *acceptable* with respect to $|\ |$ and I if, for every clause $A \leftarrow L_1, \ldots, L_n$ in $ground(P)$, and for every i with $1 \leq i \leq n$,

$$I \models L_1 \wedge \ldots \wedge L_{i-1} \text{ implies } |A| > |L_i|.$$

P is *acceptable* if it is acceptable with respect to some level mapping and some model. Loosely, a program is acceptable if there is some model for it which is

well behaved with respect to the uses of negation in the program, and with respect to which clause bodies are 'simpler' than clause heads. But the essential difference between this notion of simpler and that used in local stratification, say, is that we don't look at the whole of a clause body but only at enough of it, starting from the left, to know whether it is true or false in the model.

We will prove that for an acceptable program P, both the operators T_P and Φ_P have the same unique fixed point. We lead up to this through a sequence of definitions and results that culminate in the formal result we seek.

Assumption *For the rest of this Section, P is a general program that is acceptable with respect to the level mapping $|\ |$ and the model I.*

An acceptable program is modular, more or less in the way that the transitive closure program was: the behavior of relations in Neg_P^* does not depend on the behavior of relations not in Neg_P^*. We work with the negative part of P first. We begin with some general observations, whose verification we omit.

General Facts

1. Let v_I be the valuation corresponding to the model I. That is, for a ground atom A, $v_I(A) = true$ if and only if $I \models A$. Also let v_I^0 be v_I restricted to those ground atoms in Neg_P^*. Since I, restricted to relations in Neg_P^*, is a model for $comp(P^-)$ it follows from [1] that $T_{P^-}(v_I^0) = v_I^0$. (In fact, this is the only consequence of the assumption $I \models comp(P^-)$ that is used.)

2. The behavior of relations in Neg_P^* only depends on clauses in P^-. More precisely, suppose A is a ground atom in Neg_P^*, and let v be any valuation. Then $T_P(v)(A) = T_{P^-}(v)(A)$.

Now we introduce a partial level mapping and a pseudo-metric to handle the essentially negative part of P.

Definition7.3. If A is a ground literal in Neg_P^*, set $\|A\| = |A|$. On all other ground atoms $\|\ \|$ is not defined. The distance function d_1 is the pseudo-metric induced by this partial level mapping. Next we show a convergence result directly, without use of the Contraction Theorem.

Lemma 1. $d_1(T_P(v), v_I) = d_1(T_{P^-}(v), v_I) \leq \frac{1}{2} \cdot d_1(v, v_I)$.

PROOF. The equality, $d_1(T_P(v), v_I) = d_1(T_{P^-}(v), v_I)$, is simple, since d_1 only depends on relations in Neg_P^*, and $T_P(v)$ and $T_{P^-}(v)$ agree on such relations, by General Fact 1.

Now suppose $d(v, v_I) = 1/2^n$, so that v and v_I agree on all ground atoms A with $\|A\| < n$, but differ on a ground atom A such that $\|A\| = n$. To show $d(T_{P^-}(v), v_I) \leq \frac{1}{2} \cdot d(v, v_I)$, it is enough to show $T_{P^-}(v)$ and v_I agree on all ground atoms A with $\|A\| < n + 1$.

Let A be a ground atom with $\|A\| < n+1$, and suppose that $T_{P^-}(v)(A) \neq v_I(A)$. There are two cases that arise since $T_{P^-}(v)(A)$ can be *true* and $v_I(A)$ *false*, or the other way around. Suppose $T_{P^-}(v)(A) =$ *true* but $v_I(A) =$ *false*; the other half is somewhat easier and we omit it.

Since $\|A\|$ is defined, $A \in Neg_P^*$. Since $T_{P^-}(v)(A) =$ *true*, there is a ground instance of a clause in P^-, $A \leftarrow L_1, \ldots, L_m$, such that $v(L_1) = \ldots = v(L_m) =$ *true*. But also, using the General Facts, *false* $= v_I(A) = v_I^0(A) = T_{P^-}(v_I^0)(A)$. It follows that v_I^0 must falsify the body of clause $A \leftarrow L_1, \ldots, L_m$. Consequently for some $k \geq 1$, $v_I^0(L_1) = \ldots = v_I^0(L_{k-1}) =$ *true* and $v_I^0(L_k) =$ *false*. Then $I \models L_1 \wedge \ldots \wedge L_{k-1}$. It follows that $|A| > |L_i|$ for $i = 1, \ldots, k$, from which it further follows that $n + 1 > \|A\| > \|L_i\|$, so $\|L_i\| < n$ for $i = 1, \ldots, k$. Since $d(v, v_I) = 1/2^n$, v and v_I must agree on L_k, that is, v and v_I^0 must agree on L_k, and they do not. This contradiction establishes the Lemma. □

It follows immediately from this Lemma that any sequence v, $T_P(v)$, $T_P(T_P(v))$, ..., converges in the pseudo-metric d_1, to v_I. That is, the behavior of relations in Neg_P^* is completely determined by the program P, and agrees with the model I.

Now we turn to the relations not in Neg_P^*. This time we will find their behavior is also uniquely characterized, but need not agree with I. We will use a metric essentially consisting of two parts. One part measures how close the interpretation of relations in Neg_P^* is to their meaning in I. This is much like what we did in the previous Section. The other part concerns itself with relations not in Neg_P^*, and here we are not interested in simple closeness, but in the existence of contradictions to the 'assertions' of I and how significant they are.

Definition7.4.

1. If A is a ground literal not in Neg_P^*, set $|\!|\!|A|\!|\!| = |A|$. On all other ground atoms $|\!|\!|\ |\!|\!|$ is not defined.

2. The distance function d_2 is the pseudo-metric induced by the partial level mapping $\| \ \|$.
3. We say a valuation v *correctly asserts a ground atom* A if: $v(A) = \mathit{true} \Rightarrow I \models A$; that is, either $v(A) = \mathit{false}$ or $I \models A$.
4. A mapping ρ, on ground atoms not in Neg_P^*, is defined as follows.
   (a) Set $\rho(v) = 0$ if v correctly asserts all ground atoms $A \notin Neg_P^*$.
   (b) Otherwise, set $\rho(v) = 1/2^n$ where n is the smallest integer such that v does not correctly assert a ground atom A with $\|A\| = n$.

Now, finally, the metric we really want.

Definition7.5. The metric d_3 is given by:

$$d_3(v, w) = \max\{d_1(v, v_I), d_1(w, v_I), d_2(v, w), \rho(v), \rho(w)\}.$$

It is straightforward to show this is a pseudo-metric; we omit details. And it is a true metric since if $d_3(v, w) = 0$ then $d_1(v, v_I)$, $d_1(w, v_I)$ and $d_2(v, w)$ are all 0, so v and w must agree on ground atoms both in and not in Neg_P^*. Now, we can establish the principal result of this section.

Proposition 1. T_P is a contraction relative to the metric d_3.

PROOF. We will show $d_3(T_P(v), T_P(w)) \leq \frac{1}{2} \cdot d_3(v, w)$. Assume $d_3(v, w) = 1/2^n$, so each of $d_1(v, v_I)$, $d_1(w, v_I)$, $d_2(v, w)$, $\rho(v)$, and $\rho(w)$ are $\leq 1/2^n$. Since we have Lemma 1, both $d_1(T_P(v), v_I)$ and $d_1(T_P(w), v_I)$ are $\leq 1/2^{n+1}$. It remains to show each of $\rho(T_P(v))$, $\rho(T_P(w))$, and $d_2(T_P(v), T_P(w))$ are $\leq 1/2^{n+1}$ to finish the proof.

We begin with $\rho(T_P(v))$; the argument for $\rho(T_P(w))$ is, of course, the same. To show $\rho(T_P(v)) \leq 1/2^{n+1}$ it is necessary to show, for each ground atom $A \notin Neg_P^*$ with $\|A\| \leq n$, that $T_P(v)$ correctly asserts A. So suppose otherwise. Assume there is a ground atom A with $\|A\| \leq n$ such that $T_P(v)(A) = \mathit{true}$ but $I \not\models A$; we derive a contradiction.

Since $T_P(v)(A) = \mathit{true}$, there is a ground instance of a clause in P, $A \leftarrow L_1, \ldots, L_m$ with $v(L_1) = \ldots = v(L_m) = \mathit{true}$. Also, since I is a model for program P, and $I \not\models A$, not every literal L_i in the body of this clause can be true in I. So there is a k such that $I \models L_1, \ldots, L_{k-1}$ but $I \not\models L_k$. Note that $|L_k| < |A| \leq n$. The argument now divides into two cases.

Case 1: $L_k \in Neg_P^*$. Since $d_1(v, v_I) \leq 1/2^n$, v and v_I agree on ground literals in Neg_P^* of level $< n$, hence $\mathit{true} = v(L_k) = v_I(L_k)$, but this contradicts the fact that $I \not\models L_k$.

Case 2: $L_k \notin Neg_P^*$. (Then L_k must be a positive literal.) Since $\rho(v) \leq 1/2^n$, v must correctly assert each ground atom not in Neg_P^* whose level is $< n$; in particular, v correctly asserts L_k. But this is impossible since $v(L_k) = \mathit{true}$ but $I \not\models L_k$.

Finally we show that $d_2(T_P(v), T_P(w)) \leq 1/2^{n+1}$. This time we must show, for each ground atom $A \notin Neg_P^*$ with $\|A\| \leq n$, that $T_P(v)(A) = T_P(w)(A)$. So, for the rest of this proof, assume $\|A\| \leq n$, and $T_P(v)(A) = \mathit{true}$. We show $T_P(w)(A) = \mathit{true}$.

Since $T_P(v)(A) = \mathit{true}$, there is a ground instance, $A \leftarrow L_1, \ldots, L_m$ of a clause in P, with $v(L_1) = \ldots = v(L_m) = \mathit{true}$. Now there are two possibilities: either the body of this ground instance is true in I, or not. If it is, that is, if we have that $I \models L_1 \wedge \ldots \wedge L_m$, then $|L_i| < |A| \leq n$ for $i = 1, \ldots, m$. Since $d_2(v, w) \leq 1/2^n$, v and w must agree on literals of level $< n$, so $w(L_1) = \ldots = w(L_m) = \mathit{true}$ and so $T_P(w)(A) = \mathit{true}$.

The other alternative is that the body of $A \leftarrow L_1, \ldots, L_m$ is not true in I. We show this is not possible, completing the argument. So, suppose $I \models L_1, \ldots, L_{k-1}$ but $I \not\models L_k$. Once again, $|L_k| < |A| \leq n$. Now, just as earlier, things break into two cases.

Case 1: $L_k \in \mathit{Neg}_P^*$. Once again, since $d_1(v, v_I) \leq 1/2^n$, v and v_I agree on ground literals in Neg_P^* of level $< n$, hence $\mathit{true} = v(L_k) = v_I(L_k)$, and this contradicts the fact that $I \not\models L_k$.

Case 2: $L_k \notin \mathit{Neg}_P^*$. Since $\rho(v) \leq 1/2^n$, v must correctly assert each ground atom not in Neg_P^* whose level is $< n$; so v correctly asserts L_k. This is impossible since $v(L_k) = \mathit{true}$ but $I \not\models L_k$. □

Now using the Banach Contraction Theorem it follows that if P is acceptable with respect to some level mapping and some model, T_P has a unique fixed point, and that fixed point is reached by iterating T_P starting from any valuation, after ω steps. If we had used Φ_P instead of T_P nothing essential would have changed in the argument. Consequently there is only one three-valued fixed point as well, and so it must coincide with the unique fixed point of T_P. This is one of the results of [3].

8. CONCLUSION

In imperative programming *loop invariants* are used to guarantee that a loop behaves as desired, provided it terminates, and *variant functions* are used to guarantee termination. The problem of finding useful ones for a program that is already written is undecidable in general. Software engineers recommended that a program writer have such things in mind when designing a loop. Of course, invariants and variants can be expressed informally — they often are.

In logic programming, metrics and pseudo-metrics sometimes play a role analogous to both loop invariants and to variant functions. They can be used to show convergence in ω steps, analogous to termination. And they can be used to show convergence to a particular model, establishing program correctness. Just as with loops in imperative programming, the problem of finding a suitable metric should not begin after the program is written, but should be done during the design stage. It is likely level functions will be found more intuitive than metrics, and the definition may be quite informal. Still the essential idea is, when writing a program one should have an idea of what gets 'simpler' during query calls, and metrics are a way one can formalize this intuitive notion. We suggest, therefore, that metrics be considered as a natural tool of software engineering appropriate for the logic programming community.

The examples given in this paper illustrate some techniques for working with metrics in the context of logic programming. The intention is not to be exhaustive — after all, no general theory is presented. Rather the intention is to get people interested in metric methods — hoping that others will develop general results.

The third example, for instance, shows how the behavior of one part of a program can be established completely before moving on to other parts that depend on it. *Modularity* of program semantics is an important issue for imperative programming. To what extent can modular techniques involving metrics be developed for logic programming? We have raised the question. We hope others pursue it.

REFERENCES

1. Apt, K. R., Blair, H., and Walker, A. Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Morgan Kaufmann, 1988, pp. 89–148.
2. Apt, K. R., and Pedreschi, D. Studies in pure Prolog: termination. In *Symposium on Computational Logic* (1990), J. W. Lloyd, Ed., Springer-Verlag, pp. 150–176.
3. Apt, K. R., and Pedreschi, D. Proving termination of general Prolog programs. *Information and Computation* (Jan. 1993).
4. Batarekh, A. *Topological aspects of logic programming.* PhD thesis, Syracuse University, 1989.
5. Batarekh, A., and Subrahmanian, V. S. Topological model set deformations in logic programming. *Fundamenta Informaticae 12* (1989), 357–400.
6. Bezem, M. Characterizing termination of logic programs with level mappings. In *Proceedings of the North American Conference on Logic Programming* (1989), E. L. Lusk and R. A. Overbeek, Eds., MIT Press, pp. 69–80.
7. Bezem, M. Strong termination of logic programs. *Journal of Logic Programming 15* (1993), 79–97.
8. Cavedon, L. Continuity, consistency, and completeness properties for logic programs. In *Proceedings of the Sixth International Conference on Logic Programming* (1989), G. Levi and M. Martelli, Eds., MIT Press, pp. 571–584.
9. Cavedon, L. Acyclic logic programs and the completeness of SLDNFL-resolution. *Theoretical Computer Science 86* (1991), 81–92.
10. Fitting, M. C. A Kripke/Kleene semantics for logic programs. *Journal of Logic Programming 2* (1985), 295–312.
11. Gelfond, M., and Lifschitz, V. The stable model semantics for logic programming. In *Proc. of the Fifth Logic Programming Symposium* (Cambridge, MA, 1988), R. Kowalski and K. Bowen, Eds., MIT Press, pp. 1070–1080.
12. Lloyd, J. W. *Foundations of Logic Programming*, second ed. Springer-Verlag, 1987.
13. Sterling, L., and Shapiro, E. *The Art of Prolog.* MIT Press, 1986.
14. Van Gelder, A., Ross, K. A., and Schlipf, J. S. Unfounded sets and well-founded semantics for general logic programs. In *Proc. Seventh Symp. on Principles of Database Systems* (1988), pp. 221–230.
15. Willard, S. *General Topology.* Addison-Wesley, 1970.