Semantic Operators and Fixed-Point Theory in Logic Programming

Pascal Hitzler* and Anthony Karel Seda
Department of Mathematics, University College Cork,
Cork, Ireland
E-mail: {phitzler,aks}@ucc.ie
Web: http://maths.ucc.ie/~{pascal,seda}

Abstract

We consider rather general operators mapping valuations to (sets of) valuations in the context of the semantics of logic programming languages. This notion generalizes several of the standard operators encountered in this subject and is inspired by earlier work of M.C. Fitting. The fixed points of such operators play a fundamental role in logic programming semantics by providing standard models of logic programs and also in determining the computability properties of these standard models. We discuss some of our recent work employing topological ideas, in conjunction with order theory, to establish methods by which one can find the fixed points of the operators arising in logic programming semantics.

Keywords: logic programming, operators, denotational semantics, fixed points, supported model, topology, generalized metric.

Introduction

A common strategy in studying the semantics of logic programming systems is to associate with each program P in the system an operator F which maps valuations to (sets of) valuations and to consider its fixed points. The latter may frequently be identified with the various important models of P, for suitably chosen F, such as the supported model (given by the Clark completion semantics), the stable model, the well-founded model etc. Furthermore, since the fixed points of F are usually found by iterating on some simple valuation, one gains information about the computability properties of the fixed point from the behaviour of the (possibly transfinite) sequence of iterates and in particular whether or not the iterates close off before reaching the transfinite.

Often the operator F is monotonic, relative to some ordering on the space of valuations under consideration, and the fixed points can be found by applying the well-known Knaster-Tarski theorem or, better, Kleene's theorem if one knows that F is actually continuous. However, there are important exceptions to this which are brought about by the introduction of negation into the syntax of the programming system. A considerable amount of effort, some of it by the present authors, has therefore been expended in studying methods of a topological nature, to be used in conjunction with order, which enable one to obtain fixed points of non-monotonic operators. Indeed, our purpose in this paper is two-fold. First, we present a new and rather general formulation of the sorts of operators which are important in this subject, and show that it contains some of the standard ones within it. Second, we give a discussion and a brief review of the topological methods we have developed and which can be used to solve the problem of finding fixed points. However, in describing these methods we present some generalizations of our earlier results, and also raise some natural questions which provide the basis for ongoing and future investigations into this subject.

Preliminaries

We consider (possibly infinite) sets Π of

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

variable-free rules r of the following form

$$A_1 \vee \cdots \vee A_n \leftarrow B_{n+1} \wedge \cdots \wedge B_m \wedge \neg B_{m+1} \wedge \cdots \wedge \neg B_k,$$

where all the A_i, B_j are atoms in some first order language $\mathcal{L}$. For convenience, we may write such rules simply as $H_r \leftarrow \texttt{body}_r$, where H_r denotes the head $A_1 \vee \cdots \vee A_n$ of the rule and $\texttt{body}_r$ denotes $B_{n+1} \wedge \cdots \wedge B_m \wedge \neg B_{m+1} \wedge \cdots \wedge \neg B_k$ i.e. its body. In practice, Π might be the set of all ground instantiations of the rules or clauses in a deductive database or disjunctive logic program, see [18], but at the moment is quite general except that, as already noted, each atom in each rule contains only constant symbols and function symbols and no variable symbols. By B_Π we denote the Herbrand base of the underlying language $\mathcal{L}$, i.e. the set of all ground atoms in $\mathcal{L}$, although all we have to say here applies to completely general sets of atoms in place of B_Π. We assume we have given a set $\mathcal{T}$ of truth values containing at least the two distinguished values `false` and `true`, and also that we have truth tables for the usual connectives $\vee$, $\wedge$, $\leftarrow$ and $\neg$. By I_Π we denote the set of all *valuations* or *interpretations* $I : B_\Pi \rightarrow \mathcal{T}$. As usual, any valuation I can be extended, using the truth tables, to give a truth value in $\mathcal{T}$ to any variable-free formula in $\mathcal{L}$. It will also be convenient at times to add to the language $\mathcal{L}$ two atoms denoted by `false` and `true`, and this notation should not cause confusion with that used for truth values. Finally, we make one standing condition concerning the given truth table for the connective $\leftarrow$ in that a formula of the form $t_i \leftarrow t_i$ must always evaluate to `true` for any truth value $t_i \in \mathcal{T}$.

Consequence Operators

Given two rules r_1 and r_2 in Π, we shall say that their heads H_{r_1} and H_{r_2} are *equal* if they contain the same atoms (the order of the disjuncts is irrelevant). Given the head H_r of some rule r, we form the collection of all rules $H_r \leftarrow C_i$ with that head and form the corresponding *pseudo-rule* $H_r \leftarrow C_1 \vee C_2 \vee \ldots$, where the C_i denote the bodies of the corresponding rules; by an abuse of notation we refer to this as a pseudo-rule in Π. Finally, we assume that each valuation I in I_Π is extended to give truth values to the (possibly infinite) disjunction $C_1 \vee C_2 \vee \ldots$ of the bodies. We are now in a position to define the notion of a consequence operator, which is a mapping $F : I_\Pi \rightarrow \mathcal{P}(I_\Pi)$ mapping valuations to sets of valuations ($\mathcal{P}(I_\Pi)$ denotes the power set of I_Π).

Definition 1 A mapping $F : I_\Pi \rightarrow \mathcal{P}(I_\Pi)$ is called a *consequence operator* if for any given $I \in I_\Pi$ and $J \in F(I)$ we have
(1) for each pseudo-rule $H_r \leftarrow C_1 \vee C_2 \vee \ldots$ in Π, $J(H_r) \leftarrow I(C_1 \vee C_2 \vee \ldots)$ evaluates to `true`.

It turns out that the notion of consequence operator is very general, but it will nevertheless enable us to establish a result, Theorem 6, concerning the behaviour of these operators and of models of the underlying programs. In fact, we proceed next to show that several semantic operators known in the literature are in fact consequence operators as just defined.

Definition 2 We define an operator $F_\Pi : I_\Pi \rightarrow \mathcal{P}(I_\Pi)$ as follows. For a given $I \in I_\Pi$, a valuation J belongs to $F_\Pi(I)$ iff it satisfies:
(1) for each pseudo-rule $H_r \leftarrow C_1 \vee C_2 \vee \ldots$ in Π, we have $J(H_r) = I(C_1 \vee C_2 \vee \ldots)$,
(2) whenever $J(A) = \texttt{true}$ for an atom A, there exists a pseudo-rule $H_r \leftarrow C_1 \vee C_2 \vee \ldots$ in Π such that $A \in H_r$.

This operator F_Π is indeed a consequence operator by our standing condition on $\leftarrow$, and furthermore specializes to other known operators, and we give next some examples to substantiate this claim.

Example 3 Take $\mathcal{T}$ to contain just the two truth values `false` and `true` and work in classical two-valued logic, with `false` as default value, extending I to infinite disjunctions in the usual way so that $I(C_1 \vee C_2 \vee \ldots) = \texttt{true}$ iff $I(C_i) = \texttt{true}$ for at least one i. Then we can reformulate (1) and (2) of Definition 2 as follows.

Proposition 4 With the choices just made, we have $J \in F_\Pi(I)$ iff J satisfies:
(a) for each rule $H_r \leftarrow \texttt{body}_r$ in Π such that $I(\texttt{body}_r) = \texttt{true}$, there exists $A \in H_r$ such that $J(A) = \texttt{true}$,

(b) whenever $J(A) = \texttt{true}$, there exists a rule $H_r \leftarrow \texttt{body}_r$ in Π such that $I(\texttt{body}_r) = \texttt{true}$ and $A \in H_r$.

Proof Suppose (1) and (2) hold, and that $H_r \leftarrow \texttt{body}_r$ in Π satisfies $I(\texttt{body}_r) = \texttt{true}$. Let $H_r \leftarrow C_1 \vee C_2 \vee \ldots$ be the corresponding pseudo-rule. Then $I(C_1 \vee C_2 \vee \ldots) = \texttt{true}$. Therefore, $J(H_r) = \texttt{true}$ by (1) and hence there must be some $A \in H_r$ with $J(A) = \texttt{true}$, so that (a) holds. Now suppose that $A \in B_\Pi$ is arbitrary with $J(A) = \texttt{true}$. By (2) there is a pseudo-rule $H_r \leftarrow C_1 \vee C_2 \vee \ldots$ in Π such that $A \in H_r$. Thus, $J(H_r) = \texttt{true}$, whence $I(C_1 \vee C_2 \vee \ldots) = \texttt{true}$ so that $I(C_i) = \texttt{true}$ for some i and we obtain a rule $H_r \leftarrow C_i$ in Π satisfying (b).

Conversely, suppose that (a) and (b) hold. For (1), let $H_r \leftarrow C_1 \vee C_2 \vee \ldots$ be a pseudo-rule in Π. If every C_i is false in I, then $C_1 \vee C_2 \vee \ldots$ is false in I, and so is H_r by default. So $J(H_r) = I(C_1 \vee C_2 \vee \ldots)$. If some C_i is true in I, then by (a) there is $A \in H_r$ with $J(A) = \texttt{true}$. But then $I(C_1 \vee C_2 \vee \ldots) = \texttt{true} = J(A)$, and so (1) holds. Finally, (2) is immediate from (b).

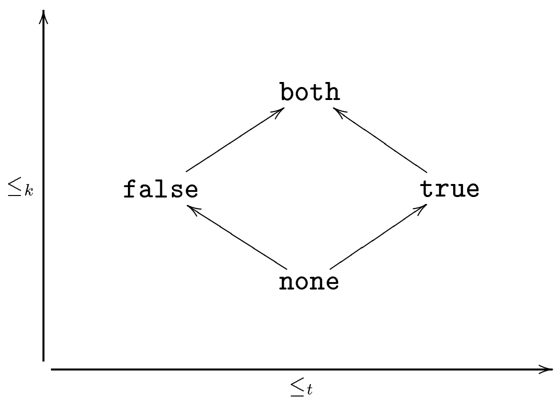
Continuing with this example, we note that, in general, a *fixed point* of a multivalued mapping $F : X \to \mathcal{P}(X)$ is an element $x \in X$ such that $x \in F(x)$; of course, if F is single-valued, this definition gives the usual meaning of a fixed point in that $F(x) = x$. Therefore, the significance of Proposition 4 is that it now follows from [8, Theorem 3.4] that $I \in F_\Pi(I)$, i.e. is a fixed point of F_Π, iff I is a *supported model* of Π in that (i) I is a model for Π, i.e. for every rule $H_r \leftarrow \texttt{body}_r$ in Π such that $\texttt{body}_r$ is true with respect to I, we have that H_r is also true with respect to I, and (ii) for every $A \in I$, there is a rule $H_r \leftarrow \texttt{body}_r$ in Π such that $\texttt{body}_r$ is true with respect to I and $A \in H_r$. Thus, any supported model of a deductive database or disjunctive program, such as the stable model of Gelfond and Lifschitz [7], can be thought of as a fixed point of F_Π. Note also that if all the rules in Π are non-disjunctive, so that we are in the context of normal logic programs P, then the operator F_Π collapses to the usual singe-step operator T_P, see [15]. ■

Example 5 Take a normal logic program P and form the set $\texttt{ground}(P)$ of all the ground clauses determined by P. For any ground atom A in $\mathcal{L}$ which does not occur in $\texttt{ground}(P)$, we add the clause $A \leftarrow \texttt{false}$. Taking Π to be the set of clauses which results from this process, we form the set of all pseudo-clauses determined by Π. We will take first the truth set $\mathcal{T}$ to contain the truth values $\texttt{false}, \texttt{true}$ and $\perp$ (undefined) and endow this set with any one of several three-valued logics such as Kleene's strong and weak three-valued logics considered in [4, 6, 9] noting that we take disjunction in pseudo-clauses as specified by the chosen three-valued logic. In the case of Kleene's strong three-valued logic, the operator F_Π as given by Definition 2 coincides with Fitting's operator Φ_P as defined in [4, 6]. Continuing in this setting but now changing the truth set $\mathcal{T}$ to the four truth values $\texttt{false}, \texttt{true}, \texttt{none}$ and $\texttt{both}$ which occur in Belnap's four-valued logic (with truth tables as defined in [6]), we obtain this time the operator Ψ_P defined in [6]. It is worth noting also that whenever one works with non-disjunctive programs (i.e. with normal logic programs), such as in this example, there is only one valuation J satisfying Definition 2 and indeed it is uniquely determined by (1) of Definition 2 ((2) of this definition is redundant in this case because every ground atom A in $\mathcal{L}$ occurs as the head of precisely one pseudo-clause). ■

It should be noted that the advantage in working with pseudo-rules or pseudo-clauses rather than with rules or clauses is that one can vary the meaning of disjunction within the pseudo-rules or pseudo-clauses according to the logic specified in $\mathcal{T}$ and thereby gain yet more flexibility. Indeed, this observation was exploited by us in [9] in order to characterize classes of programs such as the acceptable programs of [1] by suitably choosing the underlying logic. Thus, Definitions 1 and 2 give formulations ofvery general semantic operators which unify several of the important operators considered in the semantics of logic programming. The question arises, therefore, of finding the fixed points of such operators, and we consider this next.

Models and Fixed Points

Normally the sets $\mathcal{T}$ of truth sets one requires can be equipped with orderings $\leq$ in which they are complete partial orders or even Scott domains. This is the case in all the specific logics mentioned so far, and the following Hasse diagram indicates one ($\leq_k$) of the two orderings used in Belnap's logic, for example:



The corresponding structure is then inherited by the set I_Π of valuations when ordered pointwise by $I \leq J$ iff $I(A) \leq J(A)$ for all $A \in B_\Pi$, where P is a normal logic program. In the presence of such orderings on I_P, and assuming F_P is single-valued, we can apply the usual Knaster-Tarski resp. Kleene theorems to obtain (least) fixed points provided that F_P is monotonic resp. continuous. Indeed, an extension to multivalued mappings, such as F_Π, of the Knaster-Tarski theorem was given by M. Khamsi and D. Misane, and an extension of Kleene's theorem was given by us, and both of these are discussed in [11]. Both extensions, of course, require monotonicity in a suitable sense (see [11]) and are inapplicable in the presence of negation. It is precisely for this reason, as already mentioned, that one needs to consider topological methods and we turn to this issue next.

In theoretical considerations of conventional imperative and functional programming languages the use of topology is quite well-established. For example, abstract models of computation lead to the topology of observable properties of M.B. Smyth. The Scott topology is of course widely-known for its role in solving recursive domain equations and in recursive definitions of types. Also, Bukatin [2] has argued strongly that the ideal of absolutely correct software in software engineering is largely unattainable. Rather, one should work with continuous approximations to the ideal, where continuity prevents violent departure from the ideal (provided the problem at hand is not undecidable, of course), and that such ideas should be part of the software design process; to achieve this one appears to need distance functions or generalized distance functions measuring the distance between two programs. On the other hand, in logic programming such ideas are much less well-developed and are largely confined to techniques for finding fixed points and models, but see [3] and [20] and related papers.

Given that I_Π is a complete partial order, it can always be endowed with the Scott topology. Note that in the case that $\mathcal{T}$ is two-valued logic this topology coincides with the product topology $\mathbf{2}^{B_\Pi}$, where $\mathbf{2} = \{0, 1\}$ is identified with the set $\{\texttt{false}, \texttt{true}\}$ and endowed with the Scott topology. This topology suffices where monotonicity is present, such as the case of definite programs (where negation is absent from rule bodies). Nevertheless, one difference between logic programming and the other paradigms is clear: since non-monotonicity is common in logic programming, the Scott topology will often not suffice. For this reason, other topologies are needed and one which the authors have found useful, and which was studied in detail in [19], is the topology Q consisting of the product space $\mathbf{2}^{B_\Pi}$, where this time the set $\mathbf{2} = \{0, 1\}$ is endowed with the discrete topology. There are several reasons why this topology has proved useful. First, it is difficult to conceive of a topology which reflects negation which is not somehow related to Q, see the results below. Second, if a net or sequence of valuations converges in any of the distance functions we discuss briefly below, then it converges in Q. Third, if a sequence $T_P^n(I)$ of iterates converges in Q, where P is a normal logic program, then its limit is a model of P. However, in order to use these ideas in dealing with consequence operators there is a need, as we have seen by the examples, to generalize Q to allow any finite number of truth values, not just the two classical ones, and this we do next.

Let the set $\mathcal{T}$ of truth values be endowed with the discrete topology and let Π be a set of rules. The *generalized atomic topology* $\mathcal{Q}$ is defined to be the product topology on $I_\Pi = \mathcal{T}^{B_\Pi}$. We first note some observations which follow immediately from the fact that $\mathcal{Q}$ is the finite product of discrete topologies. (i) The set $\{\mathcal{G}(A,t) \mid A \in B_\Pi, t \in \mathcal{T}\}$, where $\mathcal{G}(A,t) = \{I \in I_\Pi \mid I(A) = t\}$, is a subbase of $\mathcal{Q}$. (ii) A net I_λ in I_Π converges in $\mathcal{Q}$ if and only if for every $A \in B_\Pi$ there exists some λ_0 such that $I_\lambda(A)$ is constant for all $\lambda \geq \lambda_0$. In this case, the limit of such a net I_λ is the valuation I for which $I(A)$, for each $A \in B_\Pi$, is the truth value which is eventually obtained by $I_\lambda(A)$. (iii) $\mathcal{Q}$ is a totally disconnected compact Hausdorff topology. It is second countable if B_Π is countable.

In the following, let F be a consequence operator for a non-disjunctive program and assume that F is indeed singlevalued, i.e. that $F : I_\Pi \to I_\Pi$. We define $F^0(I) = I$ for all $I \in I_\Pi$ and recursively $F^{n+1}(I) = F(F^n(I))$ for all $n \in \mathbb{N}$. We show next that convergence of iterates of F yields models of Π.

Theorem 6 Suppose that our chosen logic is such that disjunction satisfies the following (rather reasonable) condition for all $I \in I_\Pi$, for all $A \in B_\Pi$ and for all finite conjunctions C_i of literals in B_Π: $I(A \leftarrow C_1 \vee C_2 \vee \dots)$ is `true` iff $I(A \leftarrow C_i)$ is `true` for all i. Then the following statement holds: If $F^n(I)$ converges in $\mathcal{Q}$ to some $M \in I_\Pi$, then any rule in Π evaluates to `true` in $\mathcal{T}$. If, furthermore, F is continuous in $\mathcal{Q}$, then M is a fixed point of $\mathcal{T}$.

Proof For ease of notation, let $I_n = F^n(I)$ for each n and let $A \in B_\Pi$ with $M(A) = t_i$. Then we obtain $I_{k_1}(A) = t_i$ for all $k+1 \geq k_0$ for some $k_0 \in \mathbb{N}$ by convergence in $\mathcal{Q}$. Let $A \leftarrow C_1 \vee C_2 \vee \dots$ be a pseudo rule in Π. Since F is a consequence operator, we obtain that for any $k_2 > k_0$, $I_{k_2}(C_1 \vee C_2 \vee \dots)$ must have some value t_j such that $t_i \leftarrow t_j$ yields `true`. So for any C_m we have that $I_{k_2}(C_m)$ has some value t_j such that $t_i \leftarrow t_j$ yields `true`. Since C_m is a finite conjunction of ground atoms, and since I_n converges in $\mathcal{Q}$, there must therefore exist some $l_0 \in \mathbb{N}$, chosen large enough, such that for *all* $l \geq l_0$, we have that $I_l(C_m)$ evaluates to some t_j which is independent of l and such that $t_i \leftarrow t_j$ yields `true`. In particular, we obtain that $M(C_m)$ evaluates to such a t_j. Since m was chosen arbitrarily, we obtain that $M(C_m)$, for all $m \in \mathbb{N}$, evaluates to such a t_j and consequently that $M(A \leftarrow C_1 \vee C_2 \vee \dots)$ is `true` as required.

Now, if F is continuous in $\mathcal{Q}$, then we obtain $M = \lim F^{n+1}(I) = F(\lim F^n(I)) = F(M)$ as required. ■

Conclusion

Inspired by the paper [5], the authors have, in several of their own papers (see [8, 9, 10, 11, 12, 13, 14]) investigated generalized metrics, in various forms, and related fixed-point theorems, see also [16, 20]. One underlying common theme of this work is that convergence in any of these generalized metrics implies convergence in $\mathcal{Q}$. Thus, $\mathcal{Q}$ is foundational. This work is part is an initial attempt, building on [19], to investigate very general semantic operators in many valued logics, and their foundations, in relation to the denotational semantics of logic programming systems, nonmonotonic reasoning and artificial intelligence. The focus of much of this work is on the fixed points of various operators which are associated with programs written in these paradigms, since the former provide one with a semantics (the fixed-point semantics) for the latter.

References

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

[2] M. Bukatin, "Continuous Functions as Models for Programs: Mathematics and Applications", Ph.D Thesis, Brandeis University, 2000.

[3] A.P. Ferry, "Topological Characterizations for Logic Programming Semantics", Ph.D Thesis, University of Michigan, 1994.

[4] M.C. Fitting, "A Kripke-Kleene Semantics for Logic Programming", Journal of Logic Programming 2 (1985), 295–312.

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

[6] M.C. Fitting, "Fixpoint Semantics for Logic Programming: A Survey", Theoretical Computer Science, 32 pages, to appear.

[7] M. Gelfond and V. Lifschitz, "The Stable Model Semantics for Logic Programming", in: R.A. Kowalski and K.A. Bowen (Eds.), Proceedings of the Fifth International Conference and Symposium on Logic Programming, MIT Press, 1988, pp. 1070-1080.

[8] P. Hitzler and A.K. Seda, "Multivalued Mappings, Fixed-Point Theorems and Disjunctive Databases", in: A. Butterfield and S. Flynn (Eds.), Proceedings of the Third Irish Workshop on Formal Methods (IWFM'99), Electronic Workshops in Computing (eWiC), British Computer Society, pp. 1–18.

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

[10] P. Hitzler and A.K. Seda, "The Fixed-Point Theorems of Priess-Crampe and Ribenboim in Logic Programming", Proceedings of the International Conference and Workshop on Valuation Theory, University of Saskatchewan in Saskatoon, Canada, July 1999. Fields Institute Communications Series, American Mathematical Society, Providence, R.I., pp. 1–17, to appear.

[11] P. Hitzler and A.K. Seda, "Some Issues Concerning Fixed Points in Computational Logic: Quasi-Metrics, Multivalued Mappings and the Knaster-Tarski Theorem", Proceedings of the 14th Summer Conference on Topology and its Applications: Special Session on Topology in Computer Science, New York, August 1999. Topology Proceedings Vol. 24, Summer 1999, pp. 1–28, to appear.

[12] P. Hitzler and A.K. Seda, "Dislocated Topologies", Proceedings of the Slovakian Conference in Applied Mathematics 2000, Journal of Electrical Engineering, Vol. 51, No. 12/s, Slovak Academy of Sciences (2000), 3-7.

[13] P. Hitzler and A.K. Seda, "Localizing Uniquely Determined Programs", Preprint, Department of Mathematics, University College Cork, Cork, Ireland, 2000, pp. 1-16.

[14] P. Hitzler and A.K. Seda, "A New Fixed-Point Theorem for Logic Programming Semantics". Proceedings of the joint meeting of the 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI2000) and the 6th International Conference on Information Systems Analysis and Synthesis (ISAS2000), Orlando, Florida, USA, July, 2000. International Institute of Informatics and Systemics: IIIS, Vol. VII, Computer Science and Engineering Part 1, 2000, pp. 418-423.

[15] J.W. Lloyd, Foundations of Logic Programming, Berlin: Springer, 1988.

[16] S.G. Matthews, "Partial Metric Topology", Proceedings of the Eighth Summer Conference on General Topology and its Applications, Annals of the New York Academy of Sciences, Vol. 728, New York, 1994, pp. 183-197.

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

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

[19] A.K. Seda, "Topology and the Semantics of Logic Programs", Fundamenta Informaticae 24 (4) (1995), 359-386.

[20] G.-Q. Zhang and W.C. Rounds, "Complexity of Power Default Reasoning", in: Proceedings of the Twelfth Annual IEEE Symposium on Logic in Computer Science, LICS'97, Warsaw, Poland, IEEE Computer Society Press, 1997, pp. 328–339.