Minimum Model Semantics for Logic Programs with Negation-as-Failure

PANOS RONDOGIANNIS
University of Athens
and
WILLIAM W. WADGE
University of Victoria

We give a purely model-theoretic characterization of the semantics of logic programs with negation-as-failure allowed in clause bodies. In our semantics the meaning of a program is, as in the classical case, the unique *minimum* model in a program-independent ordering. We use an expanded truth domain that has an uncountable linearly ordered set of truth values between *False* (the minimum element) and *True* (the maximum), with a *Zero* element in the middle. The truth values below *Zero* are ordered like the countable ordinals. The values above *Zero* have exactly the reverse order. Negation is interpreted as reflection about *Zero* followed by a step towards *Zero*; the only truth value that remains unaffected by negation is *Zero*. We show that every program has a unique minimum model M_P, and that this model can be constructed with a T_P iteration which proceeds through the countable ordinals. Furthermore, we demonstrate that M_P can alternatively be obtained through a construction which generalizes the well-known model intersection theorem for classical logic programming. Finally, we show that by collapsing the true and false values of the infinite-valued model M_P to (the classical) *True* and *False*, we obtain a three-valued model identical to the well-founded one.

Categories and Subject Descriptors: D.1.6 [**Programming Techniques**]: Logic Programming; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*denotational semantics*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*logic and constraint programming*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*logic programming*; *nonmonotonic reasoning and belief revision*

General Terms: Theory, Languages

Additional Key Words and Phrases: Logic programming, negation-as-failure, well-founded model, infinite-valued logics

---

This work has been partially supported by the University of Athens under the project "Extensions of the Logic Programming Paradigm" (grant no. 70/4/5827) and by the Canadian Natural Sciences and Engineering Research Council (NSERC).
A preliminary version of this paper appears in the Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA 2002), Lecture Notes in Artificial Intelligence (LNAI) 2424, pages 456–467, Cosenza, Italy, September 2002.
Authors' addresses: P. Rondogiannis, Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, 157 84 Athens, Greece, e-mail: prondo@di.uoa.gr; W. W. Wadge, Department of Computer Science, University of Victoria, PO Box 3055, STN CSC, Victoria, BC, Canada V8W 3P6, e-mail: wwadge@csr.uvic.ca

1. INTRODUCTION

One of the paradoxes of logic programming is that such a small fragment of formal logic serves as such a powerful programming language. This contrast has led to many attempts to make the language more powerful by extending the fragment, but these attempts generally back-fire. The extended languages can be implemented, and are in a sense more powerful; but these extensions usually disrupt the relationship between the meaning of programs *as programs* and the meaning *as logic*. In these cases the implementation of the program-as-program can no longer be considered as computing a distinguished model of the program-as-logic. Even worse, the result of running the program may not correspond to any model at all.

The problem is illustrated by the many attempts to extend logic programming with negation (of atoms in the clause bodies). The generally accepted computational interpretation of negated atoms is *negation-as-failure*. Intuitively, a goal $\sim A$ succeeds iff the subcomputation which attempts to establish A terminates and fails. Despite its simple computational formulation, negation-as-failure proved to be extremely difficult to formalize from a semantic point of view (an overview of the existing semantic treatments is given in the next section). Moreover, the existing approaches are not purely model-theoretic in the sense that the meaning of a given program can not be computed by solely considering its set of models. This is a sharp difference from classical logic programming (without negation), in which every program has a unique *minimum* Herbrand model (which is the intersection of all its Herbrand models).

This paper presents a purely model-theoretic semantics for negation-as-failure in logic programming. In our semantics the meaning of a program is, as in the classical case, the unique minimum model in a program-independent ordering. The main contributions of the paper can be summarized as follows:

— We argue that a purely declarative semantics for logic programs with negation-as-failure should be based on an infinite-valued logic. For this purpose we introduce an expanded truth domain that has an uncountable linearly ordered set of truth values between *False* (the minimum element) and *True* (the maximum), with a *Zero* element in the middle. The truth values below *Zero* are ordered like the countable ordinals while those above *Zero* have the reverse order. This new truth domain allows us to define in a logical way the meaning of negation-as-failure and to distinguish it in a very clear manner from classical negation.

— We introduce the notions of *infinite-valued interpretation* and *infinite-valued model* for logic programs. Moreover, we define a partial ordering $\sqsubseteq_\infty$ on infinite-valued interpretations which generalizes the subset ordering of classical interpretations. We then demonstrate that every logic program that uses negation-as-failure, has a unique minimum (infinite-valued) model M_P under $\sqsubseteq_\infty$. This model can be constructed by appropriately iterating a simple T_P operator through the countable ordinals. From an algorithmic point of view, the construction of M_P proceeds in an analogous way as the *iterated least fixpoint* approach [Przymusinski 1989]. There exist however crucial differences. First and most important, the proposed approach aims at producing a unique minimum model of the program; this requirement leads to a more demanding logical setting than existing approaches and the construction of M_P is guided by the use of a family of relations on infinite-valued interpretations.

Second, the definition of T_P in the infinite-valued approach is a simple and natural extension of the corresponding well-known operator for classical logic programming; in the existing approaches the operators used are complicated by the need to keep track of the values produced at previous levels of the iteration. Of course, the proposed approach is connected to the existing ones since, as we demonstrate, if we collapse the true and false values of M_P to (classical) *True* and *False* we get the well-founded model.

— We derive an alternative characterization of the minimum model M_P which generalizes the well-known model intersection theorem of classical logic programming. To our knowledge, this is the first such result in the area of negation-as-failure (because such constructions can not be obtained if one restricts attention to either two or three-valued semantical approaches).

The rest of the paper is organized as follows: Section 2 discusses the problem of negation and gives a brief outline of the most established semantic approaches. Section 3 outlines the infinite-valued approach. Section 4 introduces infinite-valued interpretations and models, and discusses certain orderings on interpretations that will play a vital role in defining the infinite-valued semantics. The T_P operator on infinite-valued interpretations is defined in Section 5 and an important property of the operator, namely α-monotonicity, is established. In Section 6, the construction of the model M_P is presented. Section 7 establishes various properties of M_P, the most important of which is the fact that M_P is the minimum model of P under the ordering relation $\sqsubseteq_\infty$. Section 8 introduces an alternative characterization of the minimum model. Finally, Section 9 concludes the paper with discussion on certain aspects of the infinite-valued approach.

2. THE PROBLEM OF NEGATION-AS-FAILURE

The semantics of negation-as-failure is possibly the most broadly studied problem in the theory of logic programming. In this section we first discuss the problem and then present the main solutions that have been proposed until now.

2.1 The Problem

Negation-as-failure is a notion that can be described operationally in a very simple way, but whose declarative semantics has been extremely difficult to specify. This appears to be a more general phenomenon in the theory of programming languages:

> "It seems to be a general rule that programming language features and concepts which are simple operationally tend to be complex denotationally, whereas those which are simple denotationally tend to be complex operationally" [Ashcroft and Wadge 1982].

The basic idea behind negation-as-failure has as follows: suppose that we are given the goal $\leftarrow \sim A$. Now, if $\leftarrow A$ succeeds, then $\leftarrow \sim A$ fails; if $\leftarrow A$ fails finitely, then $\leftarrow \sim A$ succeeds. For example, given the program

```
p ←
r ← ~p
s ← ~q
```

the query $\leftarrow$ `r` fails because `p` succeeds, while $\leftarrow$ `s` succeeds because `q` fails.

To illustrate the problems that result from the above interpretation of negation, consider an even simpler program:

$$\texttt{works} \leftarrow \sim\texttt{tired}$$

Under the negation-as-failure rule, the meaning of the above program is captured by the model in which `tired` is *False* and `works` is *True*.

Consider on the other hand the program:

$$\texttt{tired} \leftarrow \sim\texttt{works}$$

In this case, the correct model under negation-as-failure is the one in which `works` is *False* and `tired` is *True*.

However, the above two programs have exactly the same classical models, namely:

$$\begin{aligned} M_0 &= \{(\texttt{tired}, \mathit{False}), (\texttt{works}, \mathit{True})\} \\ M_1 &= \{(\texttt{tired}, \mathit{True}), (\texttt{works}, \mathit{False})\} \\ M_2 &= \{(\texttt{tired}, \mathit{True}), (\texttt{works}, \mathit{True})\} \end{aligned}$$

We therefore have a situation in which two programs have the same *model theory* (set of models) but different computational meanings. Obviously, this implies that the computational meaning does not have a purely model-theoretic specification. In other words, one can not determine the intended model of a logic program that uses negation-as-failure by just examining its set of classical models. This is a very sharp difference from logic programming without negation in which every program has a unique minimum model.

2.2 The Existing Solutions

The first attempt to give a semantics to negation-as-failure was the so-called *program completion* approach introduced by Clark [1978]. In the completion of a program the "if" rules are replaced by "if and only if" ones and also an equality theory is added to the program (for a detailed presentation of the technique, see [Lloyd 1987]). The main problem is that the completion of a program may in certain cases be inconsistent. To circumvent the problem, Fitting [1985] considered 3-valued Herbrand models of the program completion. Later, Kunen [1987] identified a weaker version of Fitting's semantics which is recursively enumerable. However, the last two approaches do not overcome all the objections that have been raised regarding the completion (see for example the discussion in [Przymusinska and Przymusinski 1990] and in [van Gelder 1993]).

Although the program completion approach proved useful in many application domains, it has been superseded by other semantic approaches, usually termed under the name *canonical model semantics*. The basic idea of the canonical model approach is to choose among the models of a program a particular one which is presumed to be the model that the programmer had in mind. The canonical model is usually chosen among many incomparable minimal models of the program. Since (as discussed in the last subsection) the selection of the canonical model can not be performed by just examining the set of (classical) models of the program, the choice of the canonical model is inevitably driven by the syntax of the program. In the following we discuss the main semantic approaches that have resulted from this body of research.

A semantic construction that produces a single model is the so-called *stratified semantics* [Apt et al. 1988]. Informally speaking, a program is stratified if it does not contain cyclic dependencies of predicate names through negation. Every stratified logic program has a unique *perfect* model, which can be constructed in stages. As an example, consider again the program:

$$\begin{array}{l} \texttt{p} \leftarrow \\ \texttt{r} \leftarrow \sim\texttt{p} \\ \texttt{s} \leftarrow \sim\texttt{q} \end{array}$$

The basic idea in the construction of the perfect model is to rank the predicate variables according to the maximum "depth" of negation used in their defining clauses. The variables of rank 0 (like `p` and `q` above) are defined in terms of each other without use of negation. The variables of rank 1 (like `r` and `s`) are defined in terms of each other and those of rank 0, with negation applied only to variables of rank 0. Those of rank 2 are defined with negations applied only to variables of rank 1 and 0; and so on. The model can then be constructed in stages. The clauses for the rank 0 variables form a standard logic program, and its minimum model is used to assign values for the rank 0 variables. These are then treated as constants, so that the clauses for the rank 1 variables no longer have negations. The minimum model is used to assign values to the rank 1 variables, which are in turn converted to constants; and so on.

An extension of the notion of stratification is *local stratification* [Przymusinski 1988]; intuitively, in a locally stratified program, predicates may depend negatively on themselves as long as no cycles are formed when the rules of the program are instantiated. Again, every locally stratified program has a unique perfect model [Przymusinski 1988]. The construction of the perfect model can be performed in an analogous way as in the stratified case (the basic difference being that one can allow infinite countable ordinals as ranks). It is worth noting that although stratification is obviously a syntactically determinable condition, the class of locally stratified logic programs is Π^1_1-complete [Cholak and Blair 1994]. It should also be noted here that there exist some interesting cases of logic programming languages where one can establish some intermediate notion between stratification and local stratification which is powerful and decidable. For example, in temporal logic programming [Orgun 1994; Orgun and Wadge 1992] many different temporal stratification notions have been defined, and corresponding decision tests have been proposed [Zaniolo et al. 1993; Ludäscher 1998; Rondogiannis 2001].

The stratified and locally stratified semantics fail for programs in which some variables are defined (directly or indirectly) in terms of their own negations, because these variables are never ranked. For such programs we need an extra intermediate neutral truth value for certain of the negatively recursively defined variables. This approach yields the "well-founded" construction and it can be shown [van Gelder et al. 1991] that the result is indeed a model of the program. Many different constructive definitions of the well-founded model have been proposed; two of the most well-known ones are the *alternating fixpoint* [van Gelder 1989; 1993] and the *iterated least fixpoint* [Przymusinski 1989]. The well-founded model approach is compatible with stratification (it is well-known that the well-founded model of a locally stratified program coincides with its unique perfect model [van Gelder et al. 1991]).

An approach that differs in philosophy from the previous ones is the so-called *stable model semantics* [Gelfond and Lifschitz 1988]. While the "canonical model" approaches assign to a given program a unique "intended" model, the stable model semantics assigns to the program a (possibly empty) family of "intended" models. For example, the program

$$\texttt{p} \leftarrow \sim\texttt{p}$$

does not have any stable models while the program

$$\begin{aligned}\texttt{p} &\leftarrow \sim\texttt{q}\\ \texttt{q} &\leftarrow \sim\texttt{p}\end{aligned}$$

has two stable models. The stable model semantics is defined through an elegant *stability transformation* [Gelfond and Lifschitz 1988]. The connections between the stable model semantics and the previously mentioned canonical model approaches have been investigated in the literature. More specifically, it is well-known that every locally stratified program has a unique stable model which coincides with its unique perfect model [Gelfond and Lifschitz 1988]. Moreover, if a program has a two-valued well-founded model then this coincides with its unique stable model [van Gelder et al. 1991] (but the converse of this does not hold in general, see again [van Gelder et al. 1991]). Finally, as it is demonstrated in [Przymusinski 1990], the notion of stable model can be extended to a three-valued setting; then, the well-founded model can be characterized as the smallest (more precisely, the *F-least*, see [Przymusinski 1990]) three-valued stable model. The stable model approach has triggered the creation of a new promising programming paradigm, namely *answer-set programming* [Marek and Truszczynski 1999; Gelfond and Leone 2002].

It should be noted at this point that the infinite-valued approach proposed in this paper contributes to the area of the "canonical model" approaches (and not to the area of stable model semantics). In fact, as we argue in the next section, the infinite-valued semantics is the purely model-theoretic framework under which the existing canonical model approaches fall.

The discussion in this section gives only a top-level presentation of the research that has been performed regarding the semantics of negation-as-failure. For a more in-depth treatment, the interested reader should consult the many existing surveys for this area (such as for example [Apt and Bol 1994; Baral and Gelfond 1994; Przymusinska and Przymusinski 1990; Fitting 2002]).

3. THE INFINITE-VALUED APPROACH

There is a general feeling (which we share) that when one seeks a unique model, then the well-founded semantics is the right approach to negation-as-failure. There still remains however a question about its legitimacy, mainly because the well-founded model is in fact one of the *minimal* models of the program and not a *minimum* one. In other words, there is nothing that distinguishes it *as a model*.

Our goal is to remove the last doubts surrounding the well-founded model by providing a purely model-theoretic semantics (the *infinite-valued semantics*) which is compatible with the well-founded model, but in which every program with negation has a unique minimum model. In our semantics whenever two programs have the same set of infinite-valued models then they have the same minimum model.

Informally, we extend the domain of truth values and use these extra values to distinguish between ordinary negation and negation-as-failure (in fact, classical negation can be seen as strictly stronger than negation-as-failure in the sense that $\neg A$ is a more forceful statement than $\sim A$). Consider again the program:

$$
\begin{array}{l}
\mathtt{p} \leftarrow \\
\mathtt{r} \leftarrow \sim\mathtt{p} \\
\mathtt{s} \leftarrow \sim\mathtt{q}
\end{array}
$$

Under the negation-as-failure approach both `p` and `s` receive the value *True*. We would argue, however, that in some sense `p` is "truer" than `s`. Namely, `p` is true because there is a rule which says so, whereas `s` is true only because we are never obliged to make `q` true. In a sense, `s` is true only by default. Our truth domain adds a "default" truth value T_1 just below the "real" truth T_0, and (by symmetry) a weaker false value F_1 just above ("not as false as") the real false F_0. We can then understand negation-as-failure as combining ordinary negation with a weakening. Thus $\sim F_0 = T_1$ and $\sim T_0 = F_1$. Since negations can effectively be iterated, our domain requires a whole sequence $\ldots, T_3, T_2, T_1$ of weaker and weaker truth values below T_0 but above the neutral value 0; and a mirror image sequence $F_1, F_2, F_3, \ldots$ above F_0 and below 0. In fact, to capture the well-founded model in full generality, we need a T_α and a F_α for every countable ordinal α.

We show that, over this extended domain, every logic program with negation has a unique minimum model; and that in this model, if we collapse all the T_α and F_α to *True* and *False* respectively, we get the three-valued well-founded model. For the example program above, the minimum model is $\{(\mathtt{p}, T_0), (\mathtt{q}, F_0), (\mathtt{r}, F_1), (\mathtt{s}, T_1)\}$. This collapses to $\{(\mathtt{p}, \mathit{True}), (\mathtt{q}, \mathit{False}), (\mathtt{r}, \mathit{False}), (\mathtt{s}, \mathit{True})\}$, which is the well-founded model of the program.

Consider now again the program $\mathtt{works} \leftarrow\sim\mathtt{tired}$. The minimum model in this case is $\{(\mathtt{tired}, F_0), (\mathtt{works}, T_1)\}$. On the other hand, for the program $\mathtt{tired} \leftarrow\sim \mathtt{works}$ the minimum model is $\{(\mathtt{tired}, T_1), (\mathtt{works}, F_0)\}$. As it will become clearer in the next section, the minimum model of the first program is not a model of the second program, and vice-versa. Therefore, the two programs do not have the same set of infinite-valued models and the paradox identified in the previous section, disappears. Alternatively, in the infinite-valued semantics the programs $\mathtt{works} \leftarrow\sim\mathtt{tired}$ and $\mathtt{tired} \leftarrow\sim\mathtt{works}$ are no longer logically equivalent.

The proof of our minimum-model result proceeds in a manner analogous to the classical proof in the negation-free case. The main complication is that we need extra auxiliary relations to characterize the transitions between stages in the construction. This complication is unavoidable and due to the fact that in our infinite truth domain negation-as-failure is *still* antimonotonic. The approximations do converge on the least model, but not monotonically (or even anti-monotonically). Instead (speaking loosely) the values of variables with standard denotations (T_0 and F_0) are computed first, then those (T_1 and F_1) one level weaker, then those two levels weaker, and so on. We need a family of relations between models to keep track of this intricate process (whose result, nevertheless, has a simple characterization).

In other words, we consider logic programs with negation as specifying an induction process in which the values we assign to variables accumulate in stages. This idea originates from the well-known techniques for the construction of the well-founded model (see for example [Przymusinski 1989]). Recently, this idea was also made more explicit in [Denecker et al. 2001] in which the authors propose the thesis that logic programs with negation should still be considered as specifying inductive definitions that are not however monotonic[1]. In terms of the nonmonotonic induction view, our different levels of truth record the stages at which the values are assigned to variables. Expanding the truth domain allows us to give a model-theoretic treatment of nonmonotonic induction.

4. INFINITE-VALUED MODELS

In this section we define infinite-valued interpretations and infinite-valued models of programs. In the following discussion we assume familiarity with the basic notions of logic programming [Lloyd 1987]. We consider the class of normal logic programs:

Definition 4.1. A *normal program clause* is a clause whose body is a conjunction of literals. A *normal logic program* is a finite set of normal program clauses.

We follow a common practice in the area of negation, which dictates that instead of studying (finite) logic programs it is more convenient to study their (possibly infinite) *ground instantiations* [Fitting 2002]:

Definition 4.2. If P is a normal logic program, its associated *ground instantiation* P^* is constructed as follows: first, put in P^* all ground instances of members of P; second, if a clause $A \leftarrow$ with empty body occurs in P^*, replace it with $A \leftarrow \texttt{true}$; finally, if the ground atom A is not the head of any member of P^*, add $A \leftarrow \texttt{false}$.

The program P^* is in essence a (generally infinite) propositional program. In the rest of this paper, we will assume that all programs under consideration (unless otherwise stated) are of this form.

The existing approaches to the semantics of negation are either two-valued or three-valued. The two-valued approaches are based on classical logic that uses the truth values *False* and *True*. The three-valued approaches are based on a three-valued logic that uses *False*, 0 and *True*. The element 0 captures the notion of *undefined*. The truth values are ordered as: $False < 0 < True$ (see for example [Przymusinski 1989]).

The basic idea behind the proposed approach is that in order to obtain a minimum model semantics for logic programs with negation, it is necessary to consider a much more refined multiple-valued logic which is based on an infinite set of truth values, ordered as follows:

$$F_0 < F_1 < \cdots < F_\omega < \cdots < F_\alpha < \cdots < 0 < \cdots < T_\alpha < \cdots < T_\omega < \cdots < T_1 < T_0$$

Intuitively, F_0 and T_0 are the classical *False* and *True* values and 0 is the *undefined* value. The values below 0 are ordered like the countable ordinals. The values above 0 have exactly the reverse order. The intuition behind the new values is that they express different levels of truthfulness and falsity. In the following we denote by V

[1]This work was brought to our attention during the reviewing process.

the set consisting of the above truth values. A notion that will prove useful in the sequel is that of the *order* of a given truth value:

Definition 4.3. The *order* of a truth value is defined as follows: $order(T_\alpha) = \alpha$, $order(F_\alpha) = \alpha$ and $order(0) = +\infty$.

The notion of "Herbrand interpretation of a program" can now be generalized:

Definition 4.4. An (infinite-valued) interpretation I of a program P is a function from the Herbrand Base B_P of P to V.

In the rest of the paper, the term "interpretation" will mean an infinite-valued one (unless otherwise stated). As a special case of interpretation, we will use $\emptyset$ to denote the interpretation that assigns the F_0 value to all atoms of a program.

In order to define the notion of *model* of a given program, we need to extend the notion of interpretation to apply to literals, to conjunctions of literals and to the two constants `true` and `false` (for the purposes of this paper it is not actually needed to extend I to more general formulas):

Definition 4.5. Let I be an interpretation of a given program P. Then, I can be extended as follows:

—For every negative atom $\sim p$ appearing in P:

$$I(\sim p) = \begin{cases} T_{\alpha+1} & \text{if } I(p) = F_\alpha \\ F_{\alpha+1} & \text{if } I(p) = T_\alpha \\ 0 & \text{if } I(p) = 0 \end{cases}$$

—For every conjunction of literals $l_1, \ldots, l_n$ appearing as the body of a clause in P:

$$I(l_1, \ldots, l_n) = min\{I(l_1), \ldots, I(l_n)\}$$

Moreover, $I(\texttt{true}) = T_0$ and $I(\texttt{false}) = F_0$.

It is important to note that the above definition provides a purely logical characterization of what negation-as-failure is; moreover, it clarifies the difference between classical negation (which is simply reflection about 0) and negation-as-failure (which is reflection about 0 followed by a step towards 0). The operational intuition behind the above definition is that the more times a value is iterated through negation, the closer to zero it gets.

The notion of satisfiability of a clause can now be defined:

Definition 4.6. Let P be a program and I an interpretation of P. Then, I *satisfies* a clause $p \leftarrow l_1, \ldots, l_n$ of P if $I(p) \geq I(l_1, \ldots, l_n)$. Moreover, I is a *model* of P if I satisfies all clauses of P.

Given an interpretation of a program, we adopt specific notations for the set of atoms of the program that are assigned a specific truth value and for the subset of the interpretation that corresponds to a particular order:

Definition 4.7. Let P be a program, I an interpretation of P and $v \in V$. Then $I \parallel v = \{p \in B_P \mid I(p) = v\}$. Moreover, if α is a countable ordinal, then $I \sharp \alpha = \{(p, v) \in I \mid order(v) = \alpha\}$.

The following relations on interpretations will prove useful in the rest of the paper:

Definition 4.8. Let I and J be interpretations of a given program P and α be a countable ordinal. We write $I =_\alpha J$, if for all $\beta \leq \alpha$, $I \parallel T_\beta = J \parallel T_\beta$ and $I \parallel F_\beta = J \parallel F_\beta$.

Example 4.9. Let $I = \{(\mathtt{p}, T_0), (\mathtt{q}, T_1), (\mathtt{r}, T_2)\}$ and $J = \{(\mathtt{p}, T_0), (\mathtt{q}, T_1), (\mathtt{r}, F_2)\}$. Then, $I =_1 J$, but it is not the case that $I =_2 J$.

Definition 4.10. Let I and J be interpretations of a given program P and α be a countable ordinal. We write $I \sqsubset_\alpha J$, if for all $\beta < \alpha$, $I =_\beta J$ and either $I \parallel T_\alpha \subset J \parallel T_\alpha$ and $I \parallel F_\alpha \supseteq J \parallel F_\alpha$, or $I \parallel T_\alpha \subseteq J \parallel T_\alpha$ and $I \parallel F_\alpha \supset J \parallel F_\alpha$. We write $I \sqsubseteq_\alpha J$ if $I =_\alpha J$ or $I \sqsubset_\alpha J$.

Example 4.11. Consider the interpretations $I = \{(\mathtt{p}, T_0), (\mathtt{q}, T_1), (\mathtt{r}, F_2)\}$ and $J = \{(\mathtt{p}, T_0), (\mathtt{q}, T_1), (\mathtt{r}, T_2)\}$. Obviously, $I \sqsubset_2 J$.

Definition 4.12. Let I and J be interpretations of a given program P. We write $I \sqsubset_\infty J$, if there exists a countable ordinal α (that depends on I and J) such that $I \sqsubset_\alpha J$. We write $I \sqsubseteq_\infty J$ if either $I = J$ or $I \sqsubset_\infty J$.

Notice that in the above definition α depends on the interpretations I and J. More specifically, for any given I and J, α is the least countable ordinal for which $I\sharp\alpha$ is not equal to $J\sharp\alpha$. Therefore, $\sqsubset_\infty$ is not in general equal to $\sqsubset_\alpha$ for any particular fixed α.

It is easy to see that the relation $\sqsubseteq_\infty$ on the set of interpretations of a given program, is a partial order (i.e., it is reflexive, transitive and antisymmetric). On the other hand, for every countable ordinal α, the relation $\sqsubseteq_\alpha$ is a preorder (i.e., reflexive and transitive). The following lemma gives a condition related to $\sqsubseteq_\infty$ which will be used in a later section:

LEMMA 4.13. *Let I and J be two interpretations of a given program P. If for all p in P, $I(p) \leq J(p)$, then $I \sqsubseteq_\infty J$.*

PROOF. If $I = J$ then obviously $I \sqsubseteq_\infty J$. Assume $I \neq J$ and let α be the least countable ordinal such that $I\sharp\alpha \neq J\sharp\alpha$. Now, for every p in P such that $J(p) = F_\alpha$, we have $I(p) \leq F_\alpha$. However, since I and J agree on their values of order less than α, we have $I(p) = F_\alpha$. Therefore, $I \parallel F_\alpha \supseteq J \parallel F_\alpha$. On the other hand, for every p in P such that $I(p) = T_\alpha$, we have $J(p) \geq T_\alpha$. Since I and J agree on their values of order less than α, we have $J(p) = T_\alpha$. Therefore, $I \parallel T_\alpha \subseteq J \parallel T_\alpha$. Since $I\sharp\alpha \neq J\sharp\alpha$, we get $I \sqsubset_\alpha J$ which implies $I \sqsubseteq_\infty J$. □

The relation $\sqsubseteq_\infty$ will be used in the coming sections in order to define the minimum model semantics for logic programs with negation-as-failure.

Example 4.14. Consider the program P:

$$\begin{array}{l}\mathtt{p} \leftarrow \sim\!\mathtt{q}\\ \mathtt{q} \leftarrow \mathtt{false}\end{array}$$

It can easily be seen that the interpretation $M_P = \{(\mathtt{p}, T_1), (\mathtt{q}, F_0)\}$ is the least one (with respect to $\sqsubseteq_\infty$) among all infinite-valued models of P. In other words, for every infinite-valued model N of P, it is $M_P \sqsubseteq_\infty N$.

We can now define a notion of monotonicity that will be the main tool in defining the infinite-valued semantics:

Definition 4.15. Let P be a program and let α be a countable ordinal. A function Φ from the set of interpretations of P to the set of interpretations of P is called *α-monotonic* iff for all interpretations I and J of P, $I \sqsubseteq_\alpha J \Rightarrow \Phi(I) \sqsubseteq_\alpha \Phi(J)$.

Based on the notions defined above, we can now define and examine the properties of an immediate consequence operator for logic programs with negation-as-failure.

5. THE IMMEDIATE CONSEQUENCE OPERATOR

In this section we demonstrate that one can easily define a T_P operator for logic programs with negation, based on the notions developed in the last section. Moreover, we demonstrate that this operator is α-monotonic for all countable ordinals α. The α-monotonicity allows us to prove that this new T_P has a least fixpoint, for which however ω iterations are not sufficient. The procedure required for getting the least fixpoint is more subtle than that for classical logic programs, and will be described shortly.

Definition 5.1. Let P be a program and let I be an interpretation of P. The operator T_P is defined as follows:[2]

$$T_P(I)(p) = lub\{I(l_1,\ldots,l_n) \mid p \leftarrow l_1,\ldots,l_n \in P\}$$

T_P is called the *immediate consequence operator* for P.

The following lemma demonstrates that T_P is well-defined:

Lemma 5.2. *Every subset of the set V of truth values has a least upper bound.*

Proof. Let V_F and V_T be the subsets of V that correspond to the false and true values respectively. Let S be a subset of V. Consider first the case in which $S \cap V_T$ is nonempty. Then, since V_T is a reverse well-order, the subset $S \cap V_T$ must have a greatest element, which is clearly the least upper bound of S.

Now assume that $S \cap V_T$ is empty. Then, the intermediate truth value 0 is an upper bound of S. If there are no other upper bounds in V_F, then 0 is the least upper bound. But if the set of upper bounds of S in V_F is non empty, it must have a least element, because V_F is well ordered; and this least element is clearly the least upper bound of S in the whole truth domain V. □

Example 5.3. Consider the program:

```
p ← ~q
p ← ~p
q ← false
```

and the interpretation $I = \{(\mathtt{p}, T_0), (\mathtt{q}, T_1)\}$. Then, $T_P(I) = \{(\mathtt{p}, F_2), (\mathtt{q}, F_0)\}$.

[2]The notation $T_P(I)(p)$ is possibly more familiar to people having some experience with functional programming: $T_P(I)(p)$ is the value assigned to p by the interpretation $T_P(I)$.

Example 5.4. For a more demanding example consider the following infinite program:

$$\begin{array}{llll} \mathtt{p_0} \leftarrow \mathtt{false} & & \mathtt{q} \leftarrow \mathtt{p_0} \\ \mathtt{p_1} \leftarrow \sim\mathtt{p_0} & & \mathtt{q} \leftarrow \mathtt{p_1} \\ \mathtt{p_2} \leftarrow \sim\mathtt{p_1} & & \mathtt{q} \leftarrow \mathtt{p_2} \\ \mathtt{p_3} \leftarrow \sim\mathtt{p_2} & & \mathtt{q} \leftarrow \mathtt{p_3} \\ \quad \cdots & & \quad \cdots \end{array}$$

Let $I = \{(\mathtt{q}, F_0), (\mathtt{p_0}, F_0), (\mathtt{p_1}, F_1), (\mathtt{p_2}, F_2), \ldots\}$. Then, it can be easily seen that $T_P(I) = \{(\mathtt{q}, F_\omega), (\mathtt{p_0}, F_0), (\mathtt{p_1}, T_1), (\mathtt{p_2}, T_2), \ldots\}$.

One basic property of T_P is that it is α-monotonic, a property that is illustrated by the following example:

Example 5.5. Consider the program:

$$\begin{array}{l} \mathtt{p} \leftarrow \sim\mathtt{q} \\ \mathtt{q} \leftarrow \mathtt{false} \end{array}$$

Let $I = \{(\mathtt{q}, F_0), (\mathtt{p}, T_2)\}$ and $J = \{(\mathtt{q}, F_1), (\mathtt{p}, T_0)\}$. Clearly, $I \sqsubseteq_0 J$. It can easily be seen that $T_P(I) = \{(\mathtt{q}, F_0), (\mathtt{p}, T_1)\}$ and $T_P(J) = \{(\mathtt{q}, F_0), (\mathtt{p}, T_2)\}$, and obviously $T_P(I) \sqsubseteq_0 T_P(J)$.

The following lemma establishes the α-monotonicity of T_P. Notice that a similar lemma also holds for the well-founded semantics (see for example [Przymusinski 1989]).

LEMMA 5.6. *The immediate consequence operator T_P is α-monotonic, for all countable ordinals α.*

PROOF. The proof is by transfinite induction on α. Assume the lemma holds for all $\beta < \alpha$. We demonstrate that it also holds for α.

Let I, J be two interpretations of P such that $I \sqsubseteq_\alpha J$. We first establish that the values of order less that α remain intact by T_P. Since $I \sqsubseteq_\alpha J$, for all $\beta < \alpha$ we have $I \sqsubseteq_\beta J$ and $J \sqsubseteq_\beta I$. By the induction hypothesis, we have that $T_P(I) \sqsubseteq_\beta T_P(J)$ and $T_P(J) \sqsubseteq_\beta T_P(I)$, which implies that $T_P(I) =_\beta T_P(J)$, for all $\beta < \alpha$. It remains to show that $T_P(I) \parallel T_\alpha \subseteq T_P(J) \parallel T_\alpha$ and that $T_P(I) \parallel F_\alpha \supseteq T_P(J) \parallel F_\alpha$. We distinguish these two cases.

We first demonstrate that $T_P(I) \parallel T_\alpha \subseteq T_P(J) \parallel T_\alpha$. Assume that for some atom p in P it is $T_P(I)(p) = T_\alpha$. We need to show that $T_P(J)(p) = T_\alpha$. Obviously, $T_P(J)(p) \leq T_\alpha$ (if it was $T_P(J)(p) > T_\alpha$ then it would also be $T_P(I)(p) > T_\alpha$ since for all $\beta < \alpha$, $T_P(I) =_\beta T_P(J)$). Consider now the fact that $T_P(I)(p) = T_\alpha$. This implies that there exists a rule of the form $p \leftarrow q_1, \ldots, q_n, \sim w_1, \ldots, \sim w_m$ in P whose body evaluates under I to the value T_α. This means that for all q_i, $1 \leq i \leq n$, $I(q_i) \geq T_\alpha$ and for all w_i, $1 \leq i \leq m$, $I(\sim w_i) \geq T_\alpha$ (or equivalently, $I(w_i) < F_\alpha$). But then, since $I \sqsubseteq_\alpha J$, the evaluation of the body of the above rule under the interpretation J also results to the value T_α. This together with the fact that $T_P(J)(p) \leq T_\alpha$ allows us to conclude (using the definition of T_P) that $T_P(J)(p) = T_\alpha$.

It now remains to demonstrate that $T_P(I) \parallel F_\alpha \supseteq T_P(J) \parallel F_\alpha$. Assume that for some atom p in P, $T_P(J)(p) = F_\alpha$. We need to show that $T_P(I)(p) = F_\alpha$.

Obviously, $T_P(I)(p) \geq F_\alpha$ since $T_P(I) =_\beta T_P(J)$, for all $\beta < \alpha$. Now, the fact that $T_P(J)(p) = F_\alpha$ implies that for every rule for p in P, the body of the rule has a value under J that is less than or equal to F_α. Therefore, if $p \leftarrow q_1, \ldots, q_n, \sim w_1, \ldots, \sim w_m$ is one of these rules, then either there exists a q_i, $1 \leq i \leq n$, such that $J(q_i) \leq F_\alpha$, or there exists a w_i, $1 \leq i \leq m$, such that $J(\sim w_i) \leq F_\alpha$ (or equivalently $J(w_i) > T_\alpha$). But then, since $I \sqsubseteq_\alpha J$, the body of the above rule evaluates under I to a value less than or equal to F_α. Therefore, $T_P(I)(p) \leq F_\alpha$. This together with the fact that $T_P(I)(p) \geq F_\alpha$ imply that $T_P(I)(p) = F_\alpha$. $\square$

It is natural to wonder whether T_P is monotonic with respect to the relation $\sqsubseteq_\infty$. This is not the case, as the following example illustrates:

Example 5.7. Consider the program:

$$
\begin{array}{l}
\mathtt{p} \leftarrow \sim\mathtt{q} \\
\mathtt{s} \leftarrow \mathtt{p} \\
\mathtt{t} \leftarrow \sim\mathtt{s} \\
\mathtt{t} \leftarrow \mathtt{u} \\
\mathtt{u} \leftarrow \mathtt{t} \\
\mathtt{q} \leftarrow \mathtt{false}
\end{array}
$$

Consider the following interpretations: $I = \{(\mathtt{p}, T_1), (\mathtt{q}, F_0), (\mathtt{s}, F_0), (\mathtt{t}, T_1), (\mathtt{u}, F_0)\}$ and $J = \{(\mathtt{p}, T_1), (\mathtt{q}, F_0), (\mathtt{s}, F_1), (\mathtt{t}, F_1), (\mathtt{u}, F_1)\}$. Obviously, $I \sqsubseteq_\infty J$ because $I \sqsubset_0 J$. However, we have $T_P(I) = \{(\mathtt{p}, T_1), (\mathtt{q}, F_0), (\mathtt{s}, T_1), (\mathtt{t}, T_1), (\mathtt{u}, T_1)\}$ and also $T_P(J) = \{(\mathtt{p}, T_1), (\mathtt{q}, F_0), (\mathtt{s}, T_1), (\mathtt{t}, T_2), (\mathtt{u}, F_1)\}$. Clearly, $T_P(I) \not\sqsubseteq_\infty T_P(J)$.

The fact that T_P is not monotonic under $\sqsubseteq_\infty$ appears to suggest that if we want to find the least (with respect to $\sqsubseteq_\infty$) fixpoint of T_P, we should not rely on approximations based on the relation $\sqsubseteq_\infty$. The way that this minimum fixpoint can be constructed, is described in the following section.

6. CONSTRUCTION OF THE MINIMUM MODEL M_P

In this section we demonstrate how the minimum model M_P of a given program P can be constructed. The construction can informally be described as follows. As a first approximation to M_P, we start with the interpretation that assigns to every atom of P the value F_0 (as already mentioned, this interpretation is denoted by $\emptyset$). We start iterating the T_P on $\emptyset$ until both the set of atoms that have a F_0 value and the set of atoms having a T_0 value, stabilize. We keep all these atoms whose values have stabilized and reset the values of all remaining atoms to the next false value (namely F_1). The procedure is repeated until the F_1 and T_1 values stabilize, and we reset the remaining atoms to a value equal to F_2, and so on. Since the Herbrand Base of P is countable, there exists a countable ordinal δ for which this process will not produce any new atoms having F_δ or T_δ values. At this point we stop the iterations and reset all remaining atoms to the value 0. The above process is illustrated by the following example:

Example 6.1. Consider the program:

$$\begin{array}{l}\texttt{p} \leftarrow \sim\texttt{q}\\ \texttt{q} \leftarrow \sim\texttt{r}\\ \texttt{s} \leftarrow \texttt{p}\\ \texttt{s} \leftarrow \sim\texttt{s}\\ \texttt{r} \leftarrow \texttt{false}\end{array}$$

We start from the interpretation $I = \{(\texttt{p}, F_0), (\texttt{q}, F_0), (\texttt{r}, F_0), (\texttt{s}, F_0)\}$. Iterating the immediate consequence operator twice, we get in turn the following two interpretations:

$$\begin{array}{c}\{(\texttt{p}, T_1), (\texttt{q}, T_1), (\texttt{r}, F_0), (\texttt{s}, T_1)\}\\ \{(\texttt{p}, F_2), (\texttt{q}, T_1), (\texttt{r}, F_0), (\texttt{s}, T_1)\}\end{array}$$

Notice that the set of atoms having an F_0 value as well as the set of atoms having a T_0 value, have stabilized (there is only one atom having an F_0 value and none having a T_0 one). Therefore, we reset the values of all other atoms to F_1 and repeat the process until the F_1 and T_1 values converge:

$$\begin{array}{c}\{(\texttt{p}, F_1), (\texttt{q}, F_1), (\texttt{r}, F_0), (\texttt{s}, F_1)\}\\ \{(\texttt{p}, T_2), (\texttt{q}, T_1), (\texttt{r}, F_0), (\texttt{s}, T_2)\}\\ \{(\texttt{p}, F_2), (\texttt{q}, T_1), (\texttt{r}, F_0), (\texttt{s}, T_2)\}\end{array}$$

Now, the order 1 values have converged, so we reset all remaining values to F_2 and continue the iterations:

$$\begin{array}{c}\{(\texttt{p}, F_2), (\texttt{q}, T_1), (\texttt{r}, F_0), (\texttt{s}, F_2)\}\\ \{(\texttt{p}, F_2), (\texttt{q}, T_1), (\texttt{r}, F_0), (\texttt{s}, T_3)\}\\ \{(\texttt{p}, F_2), (\texttt{q}, T_1), (\texttt{r}, F_0), (\texttt{s}, F_4)\}\end{array}$$

The order 2 values have converged, and we reset the value of s to F_3:

$$\begin{array}{c}\{(\texttt{p}, F_2), (\texttt{q}, T_1), (\texttt{r}, F_0), (\texttt{s}, F_3)\}\\ \{(\texttt{p}, F_2), (\texttt{q}, T_1), (\texttt{r}, F_0), (\texttt{s}, T_4)\}\end{array}$$

The fact that we do not get any order 3 value implies that we have reached the end of the iterations. The final model results by setting the value of s to 0:

$$M_P = \{(\texttt{p}, F_2), (\texttt{q}, T_1), (\texttt{r}, F_0), (\texttt{s}, 0)\}$$

As it will be demonstrated, this is the minimum model of the program under $\sqsubseteq_\infty$.

The above notions are formalized by the definitions that follow.

Definition 6.2. Let P be a program, let I be an interpretation of P and α a countable ordinal. Moreover, assume that $I \sqsubseteq_\alpha T_P(I) \sqsubseteq_\alpha T_P^2(I) \sqsubseteq_\alpha \cdots \sqsubseteq_\alpha T_P^n(I) \sqsubseteq_\alpha \cdots$, $n < \omega$. Then, the sequence $\{T_P^n(I)\}_{n<\omega}$ is called an α-chain.

Definition 6.3. Let P be a program, let I be an interpretation of P and assume that $\{T_P^n(I)\}_{n<\omega}$ is an α-chain. Then, we define the interpretation $T_{P,\alpha}^\omega(I)$ as follows:

$$T_{P,\alpha}^\omega(I)(p) = \begin{cases} I(p) & \text{if } order(I(p)) < \alpha \\ T_\alpha & \text{if } p \in \bigcup_{n<\omega}(T_P^n(I) \parallel T_\alpha) \\ F_\alpha & \text{if } p \in \bigcap_{n<\omega}(T_P^n(I) \parallel F_\alpha) \\ F_{\alpha+1} & \text{otherwise} \end{cases}$$

The proof of the following lemma follows directly from the above definition:

LEMMA 6.4. *Let P be a program, I an interpretation of P and α a countable ordinal. Assume that $\{T_P^n(I)\}_{n<\omega}$ is an α-chain. Then, for all $n < \omega$, $T_P^n(I) \sqsubseteq_\alpha T_{P,\alpha}^\omega(I)$. Moreover, for all interpretations J such that for all $n < \omega$, $T_P^n(I) \sqsubseteq_\alpha J$, it is $T_{P,\alpha}^\omega(I) \sqsubseteq_\alpha J$.*

The following definition and lemma will be used later on to suggest that the interpretations that result during the construction of the minimum model, do not assign to variables values of the form T_α where α is a limit ordinal.

Definition 6.5. An interpretation I of a given program P is called *reasonable* if for all $(p, T_\alpha) \in I$, α is not a limit ordinal.

LEMMA 6.6. *Let P be a program and I a reasonable interpretation of P. Then, for all $n < \omega$, $T_P^n(I)$ is a reasonable interpretation of P. Moreover, if $\{T_P^n(I)\}_{n<\omega}$ is an α-chain, then $T_{P,\alpha}^\omega(I)$ is a reasonable interpretation of P.*

PROOF. The proof of the first part of the theorem is by induction on n. For $n = 0$ the result is immediate. Assume that $T_P^k(I)$ is reasonable, and consider the case of $T_P^{k+1}(I)$. Now, if (p, T_α) belongs to $T_P^{k+1}(I)$, where α is a limit ordinal, then there must exist a clause $p \leftarrow B$ in P such that $T_P^k(I)(B) = T_\alpha$. But this implies that there exists a literal l in B such that $T_P^k(I)(l) = T_\alpha$. If l is a positive literal, then this is impossible due to the induction hypothesis. If l is a negative literal, this is impossible from the interpretation of $\sim$ in Definition 4.5.

The proof of the second part of the theorem is immediate: if $(p, T_\alpha) \in T_{P,\alpha}^\omega(I)$ then (by the definition of $T_{P,\alpha}^\omega$) there exists $k < \omega$ such that $(p, T_\alpha) \in T_P^k(I)$. But this is impossible from the first part of the theorem. $\square$

We now define a sequence of interpretations of a given program P (which can be thought of as better and better approximations to the minimum model of P):

Definition 6.7. Let P be a program and let:

$$\begin{aligned} M_0 &= T_{P,0}^\omega(\emptyset) \\ M_\alpha &= T_{P,\alpha}^\omega(M_{\alpha-1}) && \text{for successor ordinal } \alpha \\ M_\alpha &= T_{P,\alpha}^\omega(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta) && \text{for limit ordinal } \alpha \end{aligned}$$

where:

$$(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta)(p) = \begin{cases} (\bigcup_{\beta<\alpha}(M_\beta \sharp \beta))(p) & \text{if this is defined} \\ F_\alpha & \text{otherwise} \end{cases}$$

The $M_0, M_1, \ldots, M_\alpha, \ldots$ are called the *approximations* to the minimum model of P.

From the above definition it is not immediately obvious that the approximations are well-defined. First, the definition of $T_{P,\alpha}^\omega$ presupposes the existence of an α-chain (for example, in the definition of M_0 one has to demonstrate that $\{T_P^n(\emptyset)\}_{n<\omega}$ is a 0-chain). Second, in the definition of $\widehat{\bigsqcup}_{\beta<\alpha} M_\beta$ above, we implicitly assume that $\bigcup_{\beta<\alpha}(M_\beta \sharp \beta)$ is a function. But in order to establish this, we have to demonstrate that the domains of the relations $M_\beta \sharp \beta$, $\beta < \alpha$, are disjoint (i.e., that no atom participates simultaneously to more than one $M_\beta \sharp \beta$). The following lemma clarifies the above situation. Notice that the lemma consists of two parts, which are proven simultaneously by transfinite induction. This is because the induction hypothesis of the second part is used in the induction step of the first part.

LEMMA 6.8. *For all countable ordinals α:*

(1) M_α *is well-defined, and*

(2) $T_P(M_\alpha) =_\alpha M_\alpha$.

PROOF. The proof is by transfinite induction on α. We distinguish three cases:

Case 1: $\alpha = 0$. In order to establish that the sequence $\{T_P^n(\emptyset)\}_{n<\omega}$ is a 0-chain, we use induction on n. For the basis case observe that $\emptyset \sqsubseteq_0 T_P(\emptyset)$. Moreover, if we assume that $T_P^n(\emptyset) \sqsubseteq_0 T_P^{n+1}(\emptyset)$, using the 0-monotonicity of T_P we get that $T_P^{n+1}(\emptyset) \sqsubseteq_0 T_P^{n+2}(\emptyset)$. Therefore, for all $n < \omega$, $T_P^n(\emptyset) \sqsubseteq_0 T_P^{n+1}(\emptyset)$. It remains to establish that $T_P(M_0) =_0 M_0$.

From Lemma 6.4, $T_P^n(\emptyset) \sqsubseteq_0 M_0$, for all n. By the 0-monotonicity of T_P, we have that for all $n < \omega$, $T_P^{n+1}(\emptyset) \sqsubseteq_0 T_P(M_0)$; moreover, obviously $\emptyset \sqsubseteq_0 T_P(M_0)$. Therefore, for all $n < \omega$, $T_P^n(\emptyset) \sqsubseteq_0 T_P(M_0)$. But then, from the second part of Lemma 6.4, $M_0 \sqsubseteq_0 T_P(M_0)$. It remains to show that $T_P(M_0) \sqsubseteq_0 M_0$. Let p be an atom in P such that $M_0(p) = F_0$. Then, for all n, $T_P^n(\emptyset)(p) = F_0$. This means that for every clause of the form $p \leftarrow B$ in P and for all $n < \omega$, $T_P^n(\emptyset)(B) = F_0$. This implies that there exists a literal l in B such that for all $n < \omega$, $T_P^n(\emptyset)(l) = F_0$ (this is easily implied by the fact that $\{T_P^n(\emptyset)\}_{n<\omega}$ is a 0-chain). Therefore, $M_0(l) = F_0$ and consequently $M_0(B) = F_0$, which shows that $T_P(M_0)(p) = F_0$. Consider on the other hand an atom p in P such that $T_P(M_0) = T_0$. Then, there exists a clause $p \leftarrow B$ in P such that $M_0(B) = T_0$. This implies that for all literals l in B, $M_0(l) = T_0$. But then there exists a k such that for all l in B and all $n \geq k$, $T_P^n(\emptyset)(l) = T_0$ (this again is implied by the fact that $\{T_P^n(\emptyset)\}_{n<\omega}$ is a 0-chain). This implies that for all $n \geq k$, $T_P^n(\emptyset)(B) = T_0$ which means that for all $n \geq k$, $T_P^{n+1}(\emptyset)(p) = T_0$. Consequently, $M_0(p) = T_0$.

Case 2: α is a limit ordinal. Then, $M_\alpha = T_{P,\alpha}^\omega(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta)$. Based on the induction hypothesis one can easily verify that the domains of the relations $M_\beta \sharp \beta$, $\beta < \alpha$, are disjoint and therefore the quantity $\widehat{\bigsqcup}_{\beta<\alpha} M_\beta$ is well-defined (intuitively, the values of order less than or equal to β in M_β have stabilized and will not change by subsequent iterations of T_P). Moreover, it is easy to see that the sequence $\{T_P^n(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta)\}_{n<\omega}$ is an α-chain (the proof is by induction on n and uses the α-monotonicity of T_P).

It remains to establish that $T_P(M_\alpha) =_\alpha M_\alpha$. We first show that $M_\alpha \sqsubseteq_\alpha T_P(M_\alpha)$. Since $\{T_P^n(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta)\}_{n<\omega}$ is an α-chain, from Lemma 6.4, $T_P^n(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta) \sqsubseteq_\alpha M_\alpha$, for all $n < \omega$. Using the α-monotonicity of T_P we get that for all $n < \omega$, $T_P^{n+1}(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta) \sqsubseteq_\alpha T_P(M_\alpha)$; moreover, $\widehat{\bigsqcup}_{\beta<\alpha} M_\beta \sqsubseteq_\alpha T_P(M_\alpha)$ (since $\widehat{\bigsqcup}_{\beta<\alpha} M_\beta \sqsubseteq_\alpha T_P(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta)$ and $T_P(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta) \sqsubseteq_\alpha T_P(M_\alpha)$). Therefore, we have that for all $n < \omega$, $T_P^n(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta) \sqsubseteq_\alpha T_P(M_\alpha)$. But then, by Lemma 6.4, $M_\alpha \sqsubseteq_\alpha T_P(M_\alpha)$. Notice that this (due to the definition of $\sqsubseteq_\alpha$) immediately implies that for all $\beta < \alpha$, $M_\alpha =_\beta T_P(M_\alpha)$.

It remains to show that $T_P(M_\alpha) \sqsubseteq_\alpha M_\alpha$. It suffices to show that $T_P(M_\alpha) \parallel$

$T_\alpha \subseteq M_\alpha \parallel T_\alpha$ and $T_P(M_\alpha) \parallel F_\alpha \supseteq M_\alpha \parallel F_\alpha$. The former statement is immediate since (by Lemma 6.6) values of the form T_α, where α is a limit ordinal, do not arise. Consider now the latter statement and let p be an atom in P such that $M_\alpha(p) = F_\alpha$. Then, by the definition of $T^\omega_{P,\alpha}$, we get that for all $n \geq 0$, $T^n_P(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta)(p) = F_\alpha$. Assume that $T_P(M_\alpha)(p) \neq F_\alpha$. Then, since $M_\alpha =_\beta T_P(M_\alpha)$ for all $\beta < \alpha$, it has to be $T_P(M_\alpha)(p) > F_\alpha$. But then this means that there exists a clause $p \leftarrow B$ in P such that $M_\alpha(B) > F_\alpha$. This implies that for every literal l in B, $M_\alpha(l) > F_\alpha$. But then, by a case analysis on the possible values that $M_\alpha(l)$ may have, one can show that there exists a k such that for all l in B and for all $n \geq k$, $T^n_P(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta)(l) > F_\alpha$. In other words, for this particular clause there exists a k such that for all $n \geq k$, $T^n_P(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta)(B) > F_\alpha$. But this implies that for all $n \geq k$, $T^{n+1}_P(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta)(p) > F_\alpha$ (contradiction). Therefore, $T_P(M_\alpha)(p) = F_\alpha$.

Case 3: α is a successor ordinal. Then, $M_\alpha = T^\omega_{P,\alpha}(M_{\alpha-1})$. As before, it is straightforward to establish that $\{T^n_P(M_{\alpha-1})\}_{n<\omega}$ is an α-chain. Moreover, demonstrating that $M_\alpha \sqsubseteq_\alpha T_P(M_\alpha)$ is performed in an entirely analogous way as in Case 2. Notice that this (due to the definition of $\sqsubseteq_\alpha$) immediately implies that for all $\beta < \alpha$, $M_\alpha =_\beta T_P(M_\alpha)$.

It remains to show that $T_P(M_\alpha) \sqsubseteq_\alpha M_\alpha$. For this, it suffices to establish that $T_P(M_\alpha) \parallel T_\alpha \subseteq M_\alpha \parallel T_\alpha$ and $T_P(M_\alpha) \parallel F_\alpha \supseteq M_\alpha \parallel F_\alpha$. Consider the former statement and let $T_P(M_\alpha)(p) = T_\alpha$, for some p in P. Then, since $M_\alpha =_\beta T_P(M_\alpha)$ for all $\beta < \alpha$, it has to be $M_\alpha(p) \leq T_\alpha$. Moreover, since $T_P(M_\alpha)(p) = T_\alpha$, there exists a clause $p \leftarrow B$ in P such that $M_\alpha(B) = T_\alpha$. This implies that for every literal l in B, $M_\alpha(l) \geq T_\alpha$. By a case analysis on the possible values that $M_\alpha(l)$ may have, one can show that there exists a k such that for all $n \geq k$, $T^n_P(M_{\alpha-1})(l) = M_\alpha(l)$. This implies that for all $n \geq k$, $T^n_P(M_{\alpha-1})(B) = M_\alpha(B) = T_\alpha$. This implies that for all $n \geq k$, $T^{n+1}_P(M_{\alpha-1})(p) \geq T_\alpha$ and therefore $M_\alpha(p) \geq T_\alpha$. Now, since $M_\alpha(p) \leq T_\alpha$, we conclude that $M_\alpha(p) = T_\alpha$.

The proof for the latter part of the statement is similar to the corresponding proof for Case 2. □

The following two lemmas are now needed in order to define the minimum model of a given program:

LEMMA 6.9. *Let P be a program. Then, there exists a countable ordinal δ such that:*

(1) $M_\delta \parallel T_\delta = \emptyset$ and $M_\delta \parallel F_\delta = \emptyset$

(2) for all $\beta < \delta$, $M_\beta \parallel T_\beta \neq \emptyset$ or $M_\beta \parallel F_\beta \neq \emptyset$

This ordinal δ is called the depth *of P*[3].

PROOF. The basic idea behind the proof is that since B_P is countable and the set of countable ordinals is uncountable, there can not exist an onto function from the former set to the latter. More specifically, consider the set S of pairs of truth values of the form (T_α, F_α), for all countable ordinals α. Consider the function F that maps each $p \in B_P$ to (T_α, F_α) if and only if $p \in M_\alpha \parallel F_\alpha \cup M_\alpha \parallel T_\alpha$. Assume

[3]The term "depth" was first used by T. Przymusinski in [Przymusinski 1989].

now that there does not exist a δ having the properties specified by the theorem. This would imply that every member of the range of F would be the map of at least one element from B_P. But this is impossible since B_P is countable while the set S is uncountable. To complete the proof, take as δ the smallest countable ordinal α such that $M_\alpha \parallel T_\alpha = \emptyset$ and $M_\alpha \parallel F_\alpha = \emptyset$. □

The following property of δ reassures us that the approximations beyond M_δ do not introduce any new truth values:

LEMMA 6.10. *Let P be a program and let δ be as in Lemma 6.9. Then, for all countable ordinals $\gamma \geq \delta$, $M_\gamma \parallel T_\gamma = \emptyset$ and $M_\gamma \parallel F_\gamma = \emptyset$.*

PROOF. (Outline) The proof is by transfinite induction on γ. The basic idea is that if either $M_\gamma \parallel T_\gamma$ (respectively $M_\gamma \parallel F_\gamma$) was nonempty, then $M_\delta \parallel T_\delta$ (respectively $M_\delta \parallel F_\delta$) would have to be nonempty. □

We can now formally define the interpretation M_P of a given program P:

$$M_P(p) = \begin{cases} M_\delta(p) & \text{if } order(M_\delta(p)) < \delta \\ 0 & \text{otherwise} \end{cases}$$

As it will be shown shortly, M_P is the least fixpoint of T_P, the minimum model of P with respect to $\sqsubseteq_\infty$, and when it is restricted to three-valued logic it coincides with the well-founded model [van Gelder et al. 1991].

7. PROPERTIES OF M_P

In this section we demonstrate that the interpretation M_P is a model of P. Moreover, we show that M_P is in fact the *minimum* model of P under $\sqsubseteq_\infty$.

THEOREM 7.1. *The interpretation M_P of a program P is a fixpoint of T_P.*

PROOF. By the definition of M_P and from Lemma 6.10, we have that for all countable ordinals α, $M_P =_\alpha M_\alpha$. Then, for all α, $T_P(M_P) =_\alpha T_P(M_\alpha) =_\alpha M_\alpha =_\alpha M_P$. Therefore, M_P is a fixpoint of T_P. □

THEOREM 7.2. *The interpretation M_P of a program P is a model of P.*

PROOF. Let $p \leftarrow B$ be a clause in P. It suffices to show that $M_P(p) \geq M_P(B)$. We have:

$$\begin{array}{rcll} M_P(p) & = & T_P(M_P)(p) & \text{(because } M_P \text{ is a fixpoint of } T_P) \\ & = & lub\{M_P(B_C) \mid (p \leftarrow B_C) \in P\} & \text{(Definition of } T_P) \\ & \geq & M_P(B) & \text{(Property of } lub) \end{array}$$

Therefore, M_P is a model of P. □

The following lemma will be used in the proof of the main theorem of this section:

LEMMA 7.3. *Let N be a model of a given program P. Then, $T_P(N) \sqsubseteq_\infty N$.*

PROOF. Since N is a model of P, then for all p in P and for all clauses of the form $p \leftarrow B$ in P, $N(p) \geq N(B)$. But then:

$$T_P(N)(p) = lub\{N(B) \mid (p \leftarrow B) \in P\} \leq N(p)$$

Therefore, we have that $T_P(N)(p) \leq N(p)$ for all p in P. Using Lemma 4.13, we get that $T_P(N) \sqsubseteq_\infty N$. □

THEOREM 7.4. *The infinite-valued model M_P is the least (with respect to $\sqsubseteq_\infty$) among all infinite-valued models of P.*

PROOF. Let N be another model of P. We demonstrate that $M_P \sqsubseteq_\infty N$. It suffices to show that for all countable ordinals α, if for all $\beta < \alpha$, $M_P =_\beta N$, then $M_P \sqsubseteq_\alpha N$. The proof is by transfinite induction on α. We distinguish three cases:

Case 1: $\alpha = 0$. We need to show that $M_P \sqsubseteq_0 N$. Now, since $M_P =_0 M_0$, it suffices to show that $M_0 \sqsubseteq_0 N$. By an inner induction, we demonstrate that for all $n < \omega$, $T_P^n(\emptyset) \sqsubseteq_0 N$. The basis case is trivial. Assume that $T_P^n(\emptyset) \sqsubseteq_0 N$. Using the 0-monotonicity of T_P, we get that $T_P^{n+1}(\emptyset) \sqsubseteq_0 T_P(N)$. From Lemma 7.3 it is $T_P(N) \sqsubseteq_\infty N$ which easily implies that $T_P(N) \sqsubseteq_0 N$. By the transitivity of $\sqsubseteq_0$ we get that $T_P^{n+1}(\emptyset) \sqsubseteq_0 N$. Therefore, for all $n < \omega$, $T_P^n(\emptyset) \sqsubseteq_0 N$. Using Lemma 6.4 we get that $M_0 \sqsubseteq_0 N$.

Case 2: α is a limit ordinal. We need to show that $M_P \sqsubseteq_\alpha N$. Since $M_P =_\alpha M_\alpha$, it suffices to show that $T^\omega_{P,\alpha}(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta) \sqsubseteq_\alpha N$. This can be demonstrated by proving that for all $n < \omega$, $T_P^n(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta) \sqsubseteq_\alpha N$. We proceed by induction on n. For $n = 0$ the result is immediate. Assume the above statement holds for n. We need to demonstrate the statement for $n+1$. Using the α-monotonicity of T_P, we get that $T_P^{n+1}(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta) \sqsubseteq_\alpha T_P(N)$. Now, it is easy to see that for all $\beta < \alpha$, $T_P(N) =_\beta N$ (this follows from the fact that for all $\beta < \alpha$, $M_\alpha =_\beta N$). From Lemma 7.3 we also have $T_P(N) \sqsubseteq_\infty N$. But then $T_P(N) \sqsubseteq_\alpha N$. Using the transitivity of $\sqsubseteq_\alpha$, we get that $T_P^{n+1}(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta) \sqsubseteq_\alpha N$. Therefore, for all $n < \omega$, $T_P^n(\widehat{\bigsqcup}_{\beta<\alpha} M_\beta) \sqsubseteq_\alpha N$. Using Lemma 6.4 we get that $M_\alpha \sqsubseteq_\alpha N$.

Case 3: α is a successor ordinal. The proof is very similar to that for Case 2. □

COROLLARY 7.5. *The infinite-valued model M_P is the least (with respect to $\sqsubseteq_\infty$) among all the fixpoints of T_P.*

PROOF. It is straightforward to show that every fixpoint of T_P is a model of P (the proof is identical to the proof of Theorem 7.2). The result follows immediately since M_P is the least model of P. □

Finally, the following theorem provides the connection between the infinite-valued semantics and the existing semantic approaches to negation:

THEOREM 7.6. *Let N_P be the interpretation that results from M_P by collapsing all true values to* True *and all false values to* False*. Then, N_P is the well-founded model of P.*

PROOF. (Outline) We consider the definition of the well-founded model given by T. Przymusinski in [Przymusinski 1989]. This construction uses three-valued interpretations but proceeds (from an algorithmic point of view) in a similar way as the construction of the infinite-valued model. More specifically, the approximations of the well-founded model are defined in [Przymusinski 1989] as follows (for a detailed explanation of the notation, see [Przymusinski 1989]):

$$\begin{array}{rcll} M_0 &=& \langle T_\emptyset, F_\emptyset \rangle & \\ M_\alpha &=& M_{\alpha-1} \cup \langle T_{M_{\alpha-1}}, F_{M_{\alpha-1}} \rangle & \text{for successor ordinal } \alpha \\ M_\alpha &=& (\bigcup_{\beta<\alpha} M_\beta) \cup \langle T_{\bigcup_{\beta<\alpha} M_\beta}, F_{\bigcup_{\beta<\alpha} M_\beta} \rangle & \text{for limit ordinal } \alpha \end{array}$$

Notice that we have slightly altered the definition of [Przymusinski 1989] for the case of limit ordinals; the new definition leads to exactly the same model (obtained in a smaller number of steps). One can now show by a transfinite induction on α that the above construction introduces at each step exactly the same true and false atoms as the infinite-valued approach. $\Box$

8. AN ALTERNATIVE CHARACTERIZATION OF THE MINIMUM MODEL

In this section we demonstrate an alternative characterization of the minimum model M_P of a program P. Actually, the proposed characterization generalizes the well-known *model intersection theorem* [vanEmden and Kowalski 1976; Lloyd 1987] that applies to classical logic programs (without negation).

The basic idea behind this new characterization can be described as follows. Let P be a given program and let $\mathcal{M}$ be the set of all its infinite-valued models. We now consider all those models in $\mathcal{M}$ whose part corresponding to T_0 values is equal to the intersection of all such parts for all models in $\mathcal{M}$, and whose part corresponding to F_0 values is equal to the union of all such parts for all models in $\mathcal{M}$. In other words, we consider all those models from $\mathcal{M}$ that have the fewest possible T_0 values and the most F_0 values. This gives us a new set S_0 of models of P (which as we demonstrate is non-empty). We repeat the above procedure starting from S_0 and now considering values of order 1. This gives us a new (non-empty) set S_1 of models of P, and so on. Finally, we demonstrate that the limit of this procedure is a set that contains a unique model, namely the minimum model M_P of P. The above (intuitive) presentation can now be formalized as follows:

Definition 8.1. Let S be a set of infinite-valued interpretations of a given program and α a countable ordinal. Then, we define $\bigwedge^\alpha S = \{(p, T_\alpha) \mid \forall M \in S, M(p) = T_\alpha\}$ and $\bigvee^\alpha S = \{(p, F_\alpha) \mid \exists M \in S, M(p) = F_\alpha\}$. Moreover, we define $\bigodot^\alpha S = (\bigwedge^\alpha S)\bigcup(\bigvee^\alpha S)$.

Let P be a program and let $\mathcal{M}$ be the set of models of P. We can now define the following sequence of sets of models of P:

$$
\begin{array}{lcll}
S_0 & = & \{M \in \mathcal{M} \mid M\sharp 0 = \bigodot^0 \mathcal{M}\} & \\
S_\alpha & = & \{M \in S_{\alpha-1} \mid M\sharp\alpha = \bigodot^\alpha S_{\alpha-1}\} & \text{for successor ordinal } \alpha \\
S_\alpha & = & \{M \in \bigcap_{\beta<\alpha} S_\beta \mid M\sharp\alpha = \bigodot^\alpha \bigcap_{\beta<\alpha} S_\beta\} & \text{for limit ordinal } \alpha
\end{array}
$$

Example 8.2. Consider again the program of Example 6.1:

```
p ← ∼q
q ← ∼r
s ← p
s ← ∼s
r ← false
```

We first construct the set S_0. We start by observing that one of the models of the program is the interpretation $\{(\mathtt{r}, F_0), (\mathtt{q}, T_1), (\mathtt{p}, F_2), (\mathtt{s}, 0)\}$. Since this model does not contain any T_0 value, we conclude that for all $M \in S_0$, $M \parallel T_0 = \emptyset$. Moreover, since the above model contains $(\mathtt{r}, F_0)$, we conclude that for all $M \in S_0$, $(\mathtt{r}, F_0) \in M$. But this implies that $(\mathtt{q}, T_1) \in M$, for all $M \in S_0$ (due to the second rule of the program and the fact that $M \parallel T_0 = \emptyset$). Using these restrictions, one can easily obtain restrictions for the values of p and s. Therefore, the set S_0 consists of the following models:

$$S_0 = \{\{(\mathtt{r}, F_0), (\mathtt{q}, T_1), (\mathtt{p}, v_{\mathtt{p}}), (\mathtt{s}, v_{\mathtt{s}})\} \mid F_2 \leq v_{\mathtt{p}} \leq T_1,\, 0 \leq v_{\mathtt{s}} \leq T_1,\, u_{\mathtt{s}} \geq v_{\mathtt{p}}\}$$

Now, observe that the model $\{(\mathtt{r}, F_0), (\mathtt{q}, T_1), (\mathtt{p}, F_2), (\mathtt{s}, 0)\}$ belongs to S_0. Since this model contains only one T_1 value, we conclude that for all $M \in S_1$, $M \parallel T_1 = \{\mathtt{q}\}$. Then, the set S_1 is the following:

$$S_1 = \{\{(\mathtt{r}, F_0), (\mathtt{q}, T_1), (\mathtt{p}, v_{\mathtt{p}}), (\mathtt{s}, v_{\mathtt{s}})\} \mid F_2 \leq v_{\mathtt{p}} \leq T_2,\, 0 \leq v_{\mathtt{s}} \leq T_2,\, u_{\mathtt{s}} \geq v_{\mathtt{p}}\}$$

Using similar arguments as above we get that the set S_2 is the following:

$$S_2 = \{\{(\mathtt{r}, F_0), (\mathtt{q}, T_1), (\mathtt{p}, F_2), (\mathtt{s}, v_{\mathtt{s}})\} \mid 0 \leq v_{\mathtt{s}} \leq T_3\}$$

In general, given a countable ordinal α, we have:

$$S_\alpha = \{\{(\mathtt{r}, F_0), (\mathtt{q}, T_1), (\mathtt{p}, F_2), (\mathtt{s}, v_{\mathtt{s}})\} \mid 0 \leq v_{\mathtt{s}} \leq T_{\alpha+1}\}$$

Observe that the model $\{(\mathtt{r}, F_0), (\mathtt{q}, T_1), (\mathtt{p}, F_2), (\mathtt{s}, 0)\}$ is the only model of the program that belongs to all S_α.

Consider now a program P and let $S_0, S_1, \ldots, S_\alpha, \ldots$ be the sequence of sets of models of P (as previously defined). We can now establish two lemmas that lead to the main theorem of this section:

LEMMA 8.3. *For all countable ordinals α, S_α is non-empty.*

PROOF. The proof is by transfinite induction on α. We distinguish three cases:

Case 1: $\alpha = 0$. Let N^* be the following interpretation:

$$N^*(p) = \begin{cases} T_0, & \text{if } \forall M \in \mathcal{M}\ (M(p) = T_0) \\ F_0, & \text{if } \exists M \in \mathcal{M}\ (M(p) = F_0) \\ T_1, & \text{otherwise} \end{cases}$$

It is easy to show (by a case analysis on the value of $N^*(p)$) that N^* is a model of program P and therefore (due to the way it has been constructed) that $N^* \in S_0$.

Case 2: α is a successor ordinal. Let $N \in S_{\alpha-1}$ be a model of P. We construct an interpretation N^* as follows:

$$N^*(p) = \begin{cases} N(p), & \text{if } \mathit{order}(N(p)) < \alpha \\ T_\alpha, & \text{if } \forall M \in S_{\alpha-1}\ (M(p) = T_\alpha) \\ F_\alpha, & \text{if } \exists M \in S_{\alpha-1}\ (M(p) = F_\alpha) \\ T_{\alpha+1}, & \text{otherwise} \end{cases}$$

We demonstrate that N^* is a model of P. Assume it is not. Then, there exists a clause $p \leftarrow B$ in P such that $N^*(p) < N^*(B)$. We perform a case analysis on the value of $N^*(p)$:

—$N^*(p) = F_\beta$, where $\beta \leq \alpha$. Then, there exists $M \in S_{\alpha-1}$ such that $M(p) = F_\beta$. Since M is a model of P, for all clauses $p \leftarrow B_C$ in P, $M(B_C) \leq F_\beta$. Consequently, for every such clause, there exists a literal l_C in B_C such that $M(l_C) \leq F_\beta$. But then, it is also $N^*(l_C) \leq F_\beta$ (by the definition of N^* and since all models in $S_{\alpha-1}$ agree on the values of order less than α). This implies that $N^*(B_C) \leq F_\beta$. Therefore, for all clauses of the form $p \leftarrow B_C$, it is $N^*(p) \geq N^*(B_C)$ (contradiction).

—$N^*(p) = T_\beta$, $\beta \leq \alpha$. Since we have assumed that $N^*(p) < N^*(B)$, it is $N^*(B) > T_\beta$. This implies that for every literal l in B, it is $N^*(l) > T_\beta$. But then, given any $M \in S_{\alpha-1}$, it is also $M(l) > T_\beta$ (since all models in $S_{\alpha-1}$ agree on the values of order less than α). Therefore, $M(B) > T_\beta$. But then, since $M(p) = T_\beta$, M is not a model of P (contradiction).

—$N^*(p) = T_{\alpha+1}$. Since we have assumed that $N^*(p) < N^*(B)$, it is $N^*(B) \geq T_\alpha$. But then, for every $l \in B$, it is $N^*(l) \geq T_\alpha$. Take now a model $M \in S_{\alpha-1}$ such that $M(p) < T_\alpha$ (such a model must exist because otherwise it would be $N^*(p) \geq T_\alpha$). Now, it is easy to see that for every literal l in B, since $N^*(l) \geq T_\alpha$, it is $M(l) = N^*(l)$. This implies that $M(B) \geq T_\alpha$. But since $M(p) < T_\alpha$, M is not a model of P (contradiction).

Therefore, N^* is a model of P. Moreover, due to the way it has been constructed, $N^* \in S_\alpha$.

Case 3: α is a limit ordinal. Let $N_0 \in S_0, N_1 \in S_1, \ldots, N_\beta \in S_\beta, \ldots, \beta < \alpha$, be models of P. We construct an interpretation N as follows:

$$N(p) = \begin{cases} (\bigcup_{\beta<\alpha}(N_\beta \sharp \beta))(p) & \text{if this is defined} \\ T_\alpha & \text{otherwise} \end{cases}$$

It is easy to see that N is a model of P and that $N \in \bigcap_{\beta<\alpha} S_\beta$. This implies that the set $\bigcap_{\beta<\alpha} S_\beta$ is non-empty (which is needed in the definition that will follow). Now we can define an interpretation N^* as follows:

$$N^*(p) = \begin{cases} N(p), & \text{if } order(N(p)) < \alpha \\ T_\alpha, & \text{if } \forall M \in \bigcap_{\beta<\alpha} S_\beta \ (M(p) = T_\alpha) \\ F_\alpha, & \text{if } \exists M \in \bigcap_{\beta<\alpha} S_\beta \ (M(p) = F_\alpha) \\ T_{\alpha+1}, & \text{otherwise} \end{cases}$$

Then, using a proof very similar to the one given for Case 2 above, we can demonstrate that N^* is a model of P. Due to the way that it has been constructed, it is obviously $N^* \in S_\alpha$. □

LEMMA 8.4. *There exists a countable ordinal δ such that if $M \in S_\delta$ then:*

(1) $M \sharp \delta = \emptyset$, and

(2) for all $\gamma < \delta$, $M \sharp \gamma \neq \emptyset$.

PROOF. Since B_P is countable, there can not be uncountably many S_α such that if $M \in S_\alpha$, $M \sharp \alpha \neq \emptyset$. Therefore, we can take δ to be the smallest ordinal that satisfies the first condition of the lemma. □

We can now demonstrate the main theorem of this section which actually states that there exists a unique model of P that belongs to all S_α:

THEOREM 8.5. *$\bigcap_\alpha S_\alpha$ is a singleton.*

PROOF. We first demonstrate that $\bigcap_\alpha S_\alpha$ can not contain more than one models. Assume that it contains two or more models, and take any two of them, say N and

M. Then, there must exist a countable ordinal, say γ, such that $N\sharp\gamma \neq M\sharp\gamma$. But then, N and M can not both belong to S_γ, and consequently they can not both belong to $\bigcap_\alpha S_\alpha$ (contradiction).

It remains to show that $\bigcap_\alpha S_\alpha$ is non-empty. By Lemma 8.4, there exists δ such that if $M \in S_\delta$ then $M\sharp\delta = \emptyset$ (and for all $\gamma < \delta$, $M\sharp\gamma \neq \emptyset$). Let $N \in S_\delta$ be a model (such a model exists because of Lemma 8.3). We can now create N^* which is identical to N but in which all atoms whose value under N has order greater than δ are set to the value 0. We demonstrate that N^* is a model of the program. Assume it is not. Consider then a clause $p \leftarrow B$ such that $N^*(p) < N^*(B)$. There are three cases:

—$N^*(p) = F_\beta$, $\beta < \delta$. Then, $N(p) = F_\beta$ and since N is a model of P, we have $N(B) \leq F_\beta$. But this easily implies that $N^*(B) \leq F_\beta$, and therefore $N^*(p) \geq N^*(B)$ (contradiction).
—$N^*(p) = T_\beta$, $\beta < \delta$. Then, $N(p) = T_\beta$ and since N is a model of P, we have $N(B) \leq T_\beta$. But this easily implies that $N^*(B) \leq T_\beta$, and therefore $N^*(p) \geq N^*(B)$ (contradiction).
—$N^*(p) = 0$. Now, if $N(p) \leq 0$ then (since N is a model) it is also $N(B) \leq 0$. This easily implies that $N^*(B) \leq 0$. Therefore, $N^*(p) \geq N^*(B)$ (contradiction). If on the other hand $N(p) > 0$ then $N(p) < T_\delta$ (because $N^*(p) = 0$). Now, since N is a model, we have $N(B) < T_\delta$. But this easily implies that $N^*(B) \leq 0$ and therefore $N^*(p) \geq N^*(B)$ (contradiction).

It is straightforward to see that (due to the way that it has been constructed) $N^* \in S_\alpha$ for all countable ordinals α. Therefore, $N^* \in \bigcap_\alpha S_\alpha$. □

Finally, we need to establish that the model M_P of P produced through the T_P operator coincides with the model produced by the above theorem:

Theorem 8.6. $\bigcap_\alpha S_\alpha = \{M_P\}$

Proof. Let N^* be the unique element of $\bigcap_\alpha S_\alpha$. Intuitively, due to the way that it has been constructed, N^* is "as compact as possible" at each level of truth values. More formally, for every model M of P and for all countable ordinals α, if for all $\beta < \alpha$, $N^* =_\beta M$, then $N^* \sqsubseteq_\alpha M$ (the proof is immediate due to the way that the sets S_α are constructed). Then, this implies that $N^* \sqsubseteq_\infty M$. Take now M to be equal to M_P. Then, $N^* \sqsubseteq_\infty M_P$ and also (from Theorem 7.4) $M_P \sqsubseteq_\infty N^*$. But since $\sqsubseteq_\infty$ is a partial order, we conclude that $N^* = M_P$. □

9. DISCUSSION

In this section we argue (at an informal level) that the proposed approach to the semantics of negation is closely related to the idea of *infinitesimals* used in Nonstandard Analysis. Actually, our truth domain can be understood as the result of extending the classical truth domain by adding a neutral zero and a whole series of *infinitesimal truth values* arbitrarily close to, but not equal to, the zero value.

Infinitesimals can be understood as values that are smaller than any "normal" real number but still nonzero. In general, each infinitesimal of order $n+1$ is considered to be infinitely smaller than any infinitesimal of order n. It should be clear now how we can place our nonstandard logic in this context. We consider negation-as-failure as ordinary negation followed by "multiplication" by an infinitesimal ϵ. T_1 and F_1 can be understood as the first order infinitesimals ϵT and ϵF, T_2 and F_2 as the second order infinitesimals $\epsilon^2 T$ and $\epsilon^2 F$, and so on.

Our approach differs from the "classical" infinitesimals in that we include infinitesimals of transfinite orders. Even in this respect, however, we are not pioneers. John Conway, in his famous book *On Numbers and Games*, constructs a field **No** extending the reals that has infinitesimals of order α for *every* ordinal α - not just, as our truth domain, for every countable ordinal. Lakoff and Nunez give a similar (less formal) construction of what they call the *granular numbers* [Lakoff and Nunez 2000]. It seems, however, that we are the first to propose infinitesimal truth values.

But why are the truth values we introduced really infinitesimals? Obviously ϵT is smaller than T, $\epsilon^2 T$ is smaller than ϵT, and so on. But why are they infinitesimals - on what grounds can we claim that ϵT, for example, is *infinitely* smaller than T. In the context of the real numbers, this question has a simple answer: ϵ is infinitely smaller than 1 because $n * \epsilon$ is smaller than 1 for *any* integer n. Unfortunately, this formulation of the notion of "infinitely smaller" has no obvious analogue in logic because there is no notion of multiplying a truth value by an integer.

There is, however, one important analogy with the classical theory of infinitesimals that emerges when we study the nonstandard ordering between models introduced. Consider the problem of comparing two hyperreals each of which is the sum of infinitesimals of different orders, i.e., the problem of determining whether or not $A < B$, where $A = a_0 + a_1 * \epsilon + a_2 * \epsilon^2 + a_3 * \epsilon^3 + \cdots$ and $B = b_0 + b_1 * \epsilon + b_2 * \epsilon^2 + b_3 * \epsilon^3 + \cdots$ (with the a_i and b_i standard reals). We first compare a_0 and b_0. If $a_0 < b_0$ then we immediately conclude that $A < B$ without examining any other coefficients. Similarly, if $a_0 > b_0$ then $A > B$. It is only in the case that $a_0 = b_0$ that the values a_1 and b_1 play a role. If they are unequal, A and B are ordered as a_1 and b_1. Only if a_1 and b_1 are *also* equal do we examine a_2 and b_2, and so on.

To see the analogy, let I and J be two of our nonstandard models and consider the problem of determining whether or not $I \sqsubseteq_\infty J$. It is not hard to see that the formal definition of $I \sqsubseteq_\infty J$ (given in Section 4) can also be characterized as follows. First, let I_0 be the finite partial model which consists of the *standard* part of I - the subset $I \parallel T_0 \cup I \parallel F_0$ of I obtained by restricting I to those variables to which I assigns standard truth values. Next, I_1 is the result of restricting I to variables assigned order 1 infinitesimal values (T_1 and F_1), and then replacing T_1 and F_1 by T_0 and F_0 (so that I_1 is also a standard interpretation). The higher "coefficients" $I_2, I_3, \ldots$ are defined in the same way. Then (stretching notation) $I = I_0 + I_1 * \epsilon + I_2 * \epsilon^2 + \cdots$ and likewise $J = J_0 + J_1 * \epsilon + J_2 * \epsilon^2 + \cdots$. Then to compare I and J we first compare the standard interpretations I_0 and J_0 using the standard relation. If $I_0 \sqsubseteq_0 J_0$, then $I \sqsubseteq_\infty J$. But if $I_0 = J_0$, then we must compare I_1 and J_1, and if they are also equal, I_2 and J_2, and so on. The analogy is actually very close, and reflects the fact that higher order truth values are negligible (equivalent to 0) compared to lower order truth values.

It seems that the concept of an infinitesimal truth value is closely related to the idea of prioritizing assertions. In constructing our minimal model the first priority is given to determining the values of the variables which receive standard truth values. This is the first approximation to the final model, and it involves essentially ignoring the contribution of negated variables because a rule with negated variables in its body can never force the variable in the head of the clause to become T_0. In fact the whole construction proceeds according to a hierarchy of priorities corresponding to degrees of infinitesimals. This suggests that infinitesimal truth could be used in other contexts which seem to require prioritizing assertions, such as for example in default logic.

ACKNOWLEDGMENTS

We wish to thank Maarten van Emden, Bruce Kapron, Christos Nomikos and John Schlipf for their comments on earlier versions of this paper.

REFERENCES

Apt, K., Blair, H., and Walker, A. 1988. Towards a Theory of Declarative Knowledge. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Morgan Kaufmann, Los Altos, CA, 89–148.

Apt, K. and Bol, R. 1994. Logic Programming and Negation: A Survey. *Journal of Logic Programming 19,20*, 9–71.

Ashcroft, E. A. and Wadge, W. W. 1982. Prescription for Semantics. *ACM Transactions on Programming Languages and Systems 4,* 2 (April), 283–294.

Baral, C. and Gelfond, M. 1994. Logic Programming and Knowledge Representation. *Journal of Logic Programming 19,* 20, 73–148.

Cholak, P. and Blair, H. 1994. The Complexity of Local Stratification. *Fundamenta Informaticae 21,* 4, 333–344.

Clark, K. 1978. Negation as Failure. In *Logic and Databases*, H. Gallaire and J. Minker, Eds. Plenum Press, New York, 293–322.

Denecker, M., Bruynooghe, M., and Marek, V. W. 2001. Logic programming revisited: Logic programs as inductive definitions. *ACM Transactions on Computational Logic 2,* 4, 623–654.

Fitting, M. 1985. A Kripke-Kleene Semantics for Logic Programs. *Journal of Logic Programming 2,* 4, 295–312.

Fitting, M. 2002. Fixpoint Semantics for Logic Programming: A Survey. *Theoretical Computer Science 278,* 1–2, 25–51.

Gelfond, M. and Leone, N. 2002. Logic Programming and Knowledge Representation - the A-Prolog perspective. *Artificial Intelligence 138,* 1–2, 3–38.

Gelfond, M. and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In *Proceedings of the Fifth Logic Programming Symposium.* MIT Press, 1070–1080.

Kunen, K. 1987. Negation in Logic Programming. *Journal of Logic Programming 4,* 4, 289–308.

Lakoff, G. and Nunez, R. 2000. *Where Mathematics comes from.* Basic Books.

Lloyd, J. 1987. *Foundations of Logic Programming.* Springer-Verlag.

Ludäscher, B. 1998. Integration of Active and Deductive Database Rules. Ph.D. thesis, Institut für Informatik, Universität Freiburg.

Marek, V. W. and Truszczynski, M. 1999. Stable Models and an Alternative Logic Programming Paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective.* Springer-Verlag, 375–398.

Orgun, M. 1994. Temporal and Modal Logic Programming. *SIGART Bulletin 5,* 3 (July).

Orgun, M. and Wadge, W. W. 1992. Towards a Unified Theory of Intensional Logic Programming. *Journal of Logic Programming 13,* 4.

Przymusinska, H. and Przymusinski, T. 1990. Semantic Issues in Deductive Databases and Logic Programs. In *Formal Techniques in Artificial Intelligence: a Source-Book*, R. Banerji, Ed. North Holland, 321–367.

PRZYMUSINSKI, T. 1988. On the Declarative Semantics of Deductive Databases and Logic Programs. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Morgan Kaufmann, Los Altos, CA, 193–216.

PRZYMUSINSKI, T. 1989. Every Logic Program has a Natural Stratification and an Iterated Fixed Point Model. In *Proceedings of the 8th Symposium on Principles of Database Systems*. ACM SIGACT-SIGMOD, 11–21.

PRZYMUSINSKI, T. 1990. The Well-Founded Semantics Coincides with the Three-Valued Stable Semantics. *Fundamenta Informaticae 13,* 4, 445–463.

RONDOGIANNIS, P. 2001. Stratified Negation in Temporal Logic Programming and the Cycle-Sum Test. *Theoretical Computer Science 254,* 1-2, 663–676.

VAN GELDER, A. 1989. The Alternating Fixpoint of Logic Programs with Negation. In *Proceedings of the 8th Symposium on Principles of Database Systems*. ACM SIGACT-SIGMOD, 11–21.

VAN GELDER, A. 1993. The Alternating Fixpoint of Logic Programs with Negation. *Journal of Computer and System Sciences 47,* 1, 185–221.

VAN GELDER, A., ROSS, K. A., AND SCHLIPF, J. S. 1991. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM 38,* 3, 620–650.

VANEMDEN, M. H. AND KOWALSKI, R. A. 1976. The Semantics of Predicate Logic as a Programming Language. *Journal of the ACM 23,* 4 (October), 733–742.

ZANIOLO, C., ARNI, N., AND ONG, K. 1993. Negation and Aggregates in Recursive Rules: the $\mathcal{LDL}++$ Approach. In *Proceedings of DOOD-93*. 204–221.

Received June 2003; revised November 2003; accepted November 2003