An Infinite-Game Semantics for Well-Founded Negation in Logic Programming ⋆

Chrysida Galanaki [a], Panos Rondogiannis [a] and William W. Wadge [b]

[a] *Department of Informatics & Telecommunications, University of Athens, Panepistimiopolis, 157 84 Athens, Greece*

[b] *Department of Computer Science, University of Victoria PO Box 3055, STN CSC, Victoria, BC, Canada V8W 3P6*

Abstract

We present an infinite-game characterization of the well-founded semantics for function-free logic programs with negation. Our game is a simple generalization of the standard game for negation-less logic programs introduced by M.H. van Emden (1986, Journal of Logic Programming, 3(1), 37-53) in which two players, the *Believer* and the *Doubter*, compete by trying to prove (respectively disprove) a query. The standard game is equivalent to the minimum Herbrand model semantics of logic programming in the sense that a query succeeds in the minimum model semantics iff the Believer has a winning strategy for the game which begins with the Doubter doubting this query. The game for programs with negation that we propose follows the same rules as the standard one, except that the players swap roles every time the play "passes through" negation. We start our investigation by establishing the *determinacy* of the new game by using some classical tools from the theory of infinite-games. Our determinacy result immediately provides a novel and purely game-theoretic characterization of the semantics of negation in logic programming. We proceed to establish the connections of the game semantics to the existing semantic approaches for logic programming with negation. For this purpose, we first define a refined version of the game that uses degrees of winning and losing for the two players. We then demonstrate that this refined game corresponds exactly to the infinite-valued minimum model semantics of negation (Rondogiannis & Wadge, 2005, ACM TOCL, 6(2), 441-467). This immediately implies that the unrefined game is equivalent to the well-founded semantics (since the infinite-valued semantics is a refinement of the well-founded semantics).

⋆ The present research is supported by a project that is co-funded by the European Social Fund & National Resources - EPEAEK II - PYTHAGORAS.

Email addresses: chrysida@di.uoa.gr (Chrysida Galanaki), prondo@di.uoa.gr (Panos Rondogiannis), wwadge@csr.uvic.ca (William W. Wadge).

1 Introduction

In this paper we give an infinite-game semantics for function-free logic programs with negation. The game we propose is very simple to describe and is radically different from the technical tools that are currently used for the investigation of the semantics of negation. In particular, the game leads to a novel proof of the fact that every logic program with negation has a "distinguished" model (ie., a model that reflects the intuitive reading of a program and which for this reason has a special status among the minimal models of the program). This fact is a consequence of the *determinacy* of the game, which can be established using purely game-theoretic arguments and in particular some of the most important results in the theory of infinite-games.

We prove that our game characterization coincides with the *well-founded semantics* [17], which is the most widely acceptable semantics for negation when one seeks a unique model of the program. In order to establish this equivalence, we use a refined version of the game which allows different degrees of winning and losing for the two players. We demonstrate that this refined game is equivalent to the infinite-valued minimum model semantics for logic programming with negation that was recently introduced by two of the authors [13,14]. The fact that the infinite-valued semantics is itself a refinement of the well-founded one, leads immediately to the conclusion that the unrefined game coincides with the well-founded semantics.

In order to make the paper entirely self-contained we have included all the necessary background for both negation in logic programming and infinite games (hoping also in this way to make the paper accessible to people from both areas). The rest of the paper is organized as follows: Section 2 introduces in an intuitive way the game for logic programs with negation. Section 3 presents the well-founded semantics of logic programming with negation; our presentation is based on our recent *infinite-valued* characterization of the well-founded model. Section 4 introduces the basic concepts and results regarding infinite games of perfect information. Section 5 defines in a formal way the game for programs with negation. In Section 6 we demonstrate that the game is actually determined. Section 7 presents a refined game that uses an infinite number of potential payoffs; this new game is also shown to be determined. Section 8 finally establishes the equivalence of our game semantics to the well-founded semantics of logic programming. The paper concludes with a discussion of related work and of possible future extensions of the game.

2 Game Semantics for Logic Programming

The starting point of our investigation is a simple game semantics for ordinary negation-less logic programming which has been known for many years [16]. Suppose we have a program P and a goal clause G; for simplicity, assume that the atoms that appear in P or G are all ground. We describe how the question, "does G succeed as a query to P" can be reduced to the question, "does Player I have a winning strategy in the game Γ_{P_G}".

The game Γ_{P_G} is a two-person infinite game of perfect information. Player I, who we will also call the Believer, believes that G will succeed and his first move is to play G, thus asserting his belief. Player II, who we will also call the Doubter, thinks G will fail. His first move is to choose one of the atoms in G which he thinks will fail on its own, and plays it, thus asserting his doubts. From then on play proceeds as follows: the Believer (who thinks the atom just played by the Doubter will in fact succeed) must play a clause in the program whose head is the atom just played; and the Doubter must, on his turn, play one of the atoms in the body of this clause.

Either player can win by making a move for which his opponent has no legal response. For the Believer, this means playing a clause with an empty body; this happens when the Doubter chose to doubt an atom for which there is a fact in the program. For the Doubter, this means choosing an atom for which there is no rule; in this case the Believer has chosen a rule with an atom in its body for which there is no evidence. Finally, we must give the Doubter an important advantage: he wins if the play goes on forever with both players always giving legal responses (for example, consider a play of the game that has goal $\leftarrow p$ and whose program consists of the single rule $p \leftarrow p$). The Doubter is considered a winner in this case since he indefinitely avoids being cornered by the Believer.

It is not hard to argue informally about the correctness of the game semantics for negation-less programs. If G actually fails, the Doubter's winning strategy is to repeatedly choose atoms which themselves fail. If G succeeds, the Believer's winning strategy is to repeatedly choose rules that are applicable, ie., for which all the atoms in the body succeed. The only subtle point is that the Believer, in choosing applicable rules, must avoid ones (like $q \leftarrow q$) which do not actually advance the game.

Once the standard game is understood in terms of the informal anthropomorphic description given above, it is not hard to see how to extend it to programs with negation. There is one new rule: when one of the players plays a literal of the form $\sim p$, his opponent *must*, on the next move, play p. And this move must then be answered by playing a clause whose head is p, and so on.

The significance of the new rule is that when a negation is encountered, the players swap roles - the Believer becomes the Doubter and vice-versa. For example, suppose that Player II, who doubts q, has just played it. Player I, who believes q, plays the clause $q \leftarrow r, \sim p$. Then Player II, who doubted q, thinks the weak link is $\sim p$, and plays it. Player I, who believes q, must believe $\sim p$, which means doubting p, and playing it. Thus I, who was a believer and believed in q, has now become a doubter, who doubts p. His opponent, who was a doubter, is now a believer (in p) and must find a rule for p to play.

The rules for winning or losing require modification. As before, any player who has no legal move loses immediately. Thus either I or II can lose if they find themselves, in the doubter's role, doubting a fact or, in the believer's role, believing without evidence. Furthermore, if the game play is infinite and after a certain point one of the players remains a doubter, he wins (this situation is similar to the corresponding one for negation-less programs). Finally, if during a play the two players swap roles infinitely often, the result is a tie; intuitively, none of the players has managed to get an advantage during this play, since they both engaged in circularities through negation.

As we will demonstrate in the coming sections, the game we have just described is equivalent to the well-founded semantics of negation. The well-founded semantics is based on a three-valued logic, namely a logic that uses the truth values *False*, 0 and *True*. Intuitively, we need to demonstrate that an atom has value *True* (respectively *False*) in the well-founded model iff Player I (respectively Player II) has a winning strategy in the corresponding game. Additionally, we have to show that the value 0 corresponds to the case where the best choice for both players is to lead the game to a tie. Establishing the equivalence that we just described is not straightforward. The reason is that (as we are going to see) the well-founded model is constructed in stages, and the truth values that are introduced in different stages can be thought of as having different "strengths". On the other hand, the game we have described does not have any notion of different levels of winning or losing. Therefore, in order to establish the equivalence it would be convenient if we had on the one hand a refinement of the well-founded model in which the strengths of truth values are as explicit as possible and on the other hand a refinement of the game that uses different degrees of winning and losing.

We have recently introduced a characterization of the well-founded model that captures in a logical way this notion of different strengths of truth values [13,14]. More specifically, the *infinite-valued semantics* introduced in [13,14] is a refinement of the well-founded semantics and it uses instead an infinite number of truth values ordered as follows:

$$F_0 < F_1 < F_2 < \cdots < 0 < \cdots < T_2 < T_1 < T_0$$

Inspired by this semantics, we define a refined game which supports different

degrees of winning and losing. We then demonstrate that this new game is equivalent to the infinite-valued semantics. As it is demonstrated in [13,14], if we collapse the F_i and the T_i values of the minimum infinite-valued model to classical *False* and *True* respectively, we get the well-founded model. This immediately implies that the initial (unrefined) game is equivalent to the well-founded semantics.

3 The Well-Founded Semantics of Logic Programs

The problem of extending logic programming with negation, is probably the most broadly studied topic in the area of logic programming. 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. For example, given the program

$$p \leftarrow$$

$$r \leftarrow \sim p$$

$$s \leftarrow \sim q$$

the query $\leftarrow r$ fails because p succeeds, while the query $\leftarrow s$ succeeds because q fails. There have been many proposals for assigning a formal semantics to the above intuitive operational notion (for example the references [1,3,11,5] provide nice surveys for this area). After many years of research, it appears that the most widely acceptable approaches to the semantics of negation-as-failure are the *well-founded semantics* [17] and the *stable model semantics* [6]. The former approach provides a unique "distinguished" model of the program while the latter allows for the possibility of zero, one or many models. In this paper we will focus on the well-founded semantics.

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

which are in turn converted to constants; and so on. There exist however programs in which some atoms are defined (directly or indirectly) in terms of their own negations. In the case of such circularities it is possible that some of the atoms can not be ranked. For such atoms we need an extra intermediate neutral truth value denoted by 0. For example, the program that consists of the unique clause $p \leftarrow\sim p$ has as its well-founded model the interpretation $\{(p, 0)\}$.

We now proceed to formalize the above notions. We start by defining the syntax of the logic programming language that we consider.

A *general logic program* (or simply a *logic program*) is a *finite* set of rules of the form:

$$p \leftarrow q_0, \ldots, q_{n-1}, \sim r_0, \ldots, \sim r_{m-1}$$

where the q_i's, the r_i's and p are propositional atoms.

A remark is in order. It is a common practice in the area of negation in logic programming to study propositional programs instead of first-order ones. This is because every first-order logic program can be instantiated into a (possibly infinite) propositional program [5]. It is also a common practice in many cases to deal with *finite* propositional programs in order to avoid unnecessary complications in the proofs. The results that are obtained for finite programs can usually be lifted to the infinite case with some notational overhead. For these reasons we assume that the programs that we are considering are propositional and finite. In the concluding section we outline how our results could be extended to the infinite case. Nevertheless, it should be noted that our results apply directly to Datalog [15], the function-free subset of logic programming that has significant applications in the area of deductive databases.

Rules of the above form are called *clauses*. The symbol p is called the *head* of the clause, the q_i's are called *positive literals* and the $\sim r_i$'s *negative literals*. In the above clause, the part on the right of the $\leftarrow$ constitutes the *body* of the clause. A clause with empty body is usually termed a *unit clause* or *fact*. Finally, a *goal clause* is a formula of the form $\leftarrow p$. Given a logic program P and a goal G, we write P_G for $P \cup \{G\}$. We write $literals(P_G)$ for the set of all literals that appear in the bodies of clauses of P and in the goal clause G; similarly, we write $negvars(P)$ for the set of all propositional atoms, that appear in negative literals in the bodies of clauses of P.

As we have already mentioned, the well-founded semantics is a three-valued approach: certain atoms in the well-founded model will be assigned the value *True*, others the value *False*, and the remaining the value 0. In other words, the well-founded semantics is based on a three-valued logic whose truth values are ordered as: $False < 0 < True$ (see for example [12]).

In the following we will give a description of the well-founded model based on a recent characterization that we have derived [14] (and which will prove very convenient for our purposes). The basic idea behind this characterization is that in order to obtain a purely model-theoretic description of the well-founded semantics, 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:

$$F_0 < F_1 < F_2 < \cdots < 0 < \cdots < T_2 < T_1 < T_0$$

Intuitively, F_0 and T_0 correspond to the classical *False* and *True* values respectively. The values below 0 are ordered like the natural numbers. 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 that correspond to the different levels in the construction of the well-founded model. In the following we denote by V the set consisting of the above truth values. It should be noted that if we were dealing with infinite propositional programs then we would need to extend the truth domain to contain a T_α and a F_α for every countable ordinal α (see [14] for details). A notion that will prove useful in the sequel is that of the *order* of a given truth value:

Definition 1 *The* order *of a truth value is defined as follows:* $order(T_n) = n$, $order(F_n) = n$ *and* $order(0) = +\infty$.

Notice that in the above definition the order of 0 is taken to be $+\infty$ simply because 0 is approximated from below and from above by values of all possible finite orders.

The Herbrand Base B_P of a logic program P consists of the set of all propositional symbols that appear in the program. The notion of "interpretation of a program" can now be defined:

Definition 2 *An (infinite-valued) interpretation I of a program P is a function from B_P to V.*

As a special case of interpretation, we will use $\emptyset$ to denote the interpretation that assigns the F_0 value to all propositional 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 and to conjunctions of literals:

Definition 3 *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:*

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

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

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

Notice that in the part of the definition concerning the conjunction of literals, we assume that when the conjunction is empty, then the value is by default equal to T_0.

The above definition provides a purely logical characterization of negation-as-failure; additionally, 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 *Let P be a program and I an interpretation. 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.*

The following two definitions will be needed:

Definition 5 *Let P be a program, I an interpretation of P, $v \in V$ and $n < \omega$. Then $I \parallel v = \{p \in B_P \mid I(p) = v\}$ and $I \sharp n = \{(p, v) \in I \mid order(v) = n\}$.*

Definition 6 *Let I and J be interpretations of a given program P and let $k \in \omega$. We write $I =_k J$, if for all $n \leq k$, $I \parallel T_n = J \parallel T_n$ and $I \parallel F_n = J \parallel F_n$. We write $I \sqsubset_k J$, if for all $n < k$, $I =_n J$ and either $I \parallel T_k \subset J \parallel T_k$ and $I \parallel F_k \supseteq J \parallel F_k$, or $I \parallel T_k \subseteq J \parallel T_k$ and $I \parallel F_k \supset J \parallel F_k$. We write $I \sqsubseteq_k J$ if $I =_k J$ or $I \sqsubset_k J$.*

The infinite-valued semantics of logic programs with negation is defined with the use of an appropriate operator:

Definition 7 *Let P be a program and let I be an interpretation of P. The*

operator T_P is defined as follows:

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

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

The construction of the minimum model M_P of a given program P can informally be described as follows. As a first approximation to M_P, we start with $\emptyset$, ie., with the interpretation that assigns to every atom the value F_0. We start iterating 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. The atoms that will not get a value through this procedure, get the value 0 at the end. The above process is illustrated by the following example:

Example 8 *Consider the program:*

$p \leftarrow \sim q$

$q \leftarrow \sim r$

$s \leftarrow p$

$s \leftarrow \sim s$

We start from the interpretation $I = \{(p, F_0), (q, F_0), (r, F_0), (s, F_0)\}$. Iterating T_P twice, we get the interpretation $\{(p, F_2), (q, T_1), (r, F_0), (s, T_1)\}$ in which the order 0 values have stabilized. We reset the values of all other atoms to F_1 getting the interpretation $\{(p, F_1), (q, F_1), (r, F_0), (s, F_1)\}$. We iterate T_P two more times and we get the interpretation $\{(p, F_2), (q, T_1), (r, F_0), (s, T_2)\}$ in which the order 1 values have converged. We reset all remaining values to F_2 getting the interpretation $\{(p, F_2), (q, T_1), (r, F_0), (s, F_2)\}$. After two more iterations of T_P we get $\{(p, F_2), (q, T_1), (r, F_0), (s, F_4)\}$ in which the order 2 values have stabilized. We reset s to F_3 getting $\{(p, F_2), (q, T_1), (r, F_0), (s, F_3)\}$. We iterate T_P once getting $\{(p, F_2), (q, T_1), (r, F_0), (s, T_4)\}$. We see that there is no value of order 3 in this interpretation. This means that s can not get a value of order 3 (nor of any other finite order); therefore, we can reset it to 0. Therefore, the final model is $M_P = \{(p, F_2), (q, T_1), (r, F_0), (s, 0)\}$.

We can now formalize the above notions:

Definition 9 *Let P be a program, let I be an interpretation of P and $k \in \omega$. Moreover, assume that $I \sqsubseteq_k T_P(I) \sqsubseteq_k T_P^2(I) \sqsubseteq_k \cdots \sqsubseteq_k T_P^n(I) \sqsubseteq_k \cdots$. Then, the sequence $\{T_P^n(I)\}_{n<\omega}$ is called a k-chain.*

Definition 10 *Let P be a program, let I be an interpretation of P and assume that $\{T_P^n(I)\}_{n<\omega}$ is a k-chain. Then, we define the interpretation $T_{P,k}^{\omega}(I)$ as follows:*

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

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

Definition 11 *Let P be a program and let:*

$$M_0 = T_{P,0}^{\omega}(\emptyset)$$
$$M_k = T_{P,k}^{\omega}(M_{k-1}) \text{ for } k > 0$$

Finally, define:

$$M_P(p) = \begin{cases} (\bigcup_{k<\omega}(M_k \sharp k))(p), & \text{if this is defined} \\ 0, & \text{otherwise} \end{cases}$$

The $M_0, M_1, \ldots, M_k, \ldots$ are called the approximations *of M_P.*

In [14] it is demonstrated that the M_k in the above definition are well-defined (one needs to demonstrate that there is a k-chain involved when we use the $T_{P,k}^{\omega}$ operator). Moreover, in the same reference it is demonstrated that the following theorems hold for M_P:

Theorem 12 *For every program P, the interpretation M_P is its unique minimum infinite-valued model under an ordering relation that is independent of the syntax of the program.*

Theorem 13 *Let P be a program and let N_P be the interpretation that results from M_P by collapsing all the T_k values to True and all the F_k values to False. Then, N_P is the well-founded model of P.*

The first theorem establishes a syntax-independent characterization of the semantics of logic programs with negation. The second theorem provides a direct connection of the infinite-valued model to the well-founded one. As an

example, the well-founded model of the program in Example 8 is equal to $\{(p, F), (q, T), (r, F), (s, 0)\}$.

4 Infinite Games of Perfect Information

The semantics that we develop in this paper is based on the so-called *infinite games of perfect information* (or *PI-games* for short) [7]. The games will take place between two players that we will call *Player I* and *Player II*. A PI game is one in which there is no hidden information: both players know all the moves that have been played so far, and there are no simultaneous moves. The games are infinite in the sense that they do not terminate at a finite stage and therefore in order to derive the outcome of a play it may be necessary to examine its entire history.

Before defining PI-games in a formal way, we need to introduce some notation. Sequences (finite or infinite in length) will usually be denoted by s or x. A finite sequence of length k will be denoted by $\langle s_0, s_1, \ldots, s_{k-1}\rangle$ and the empty sequence by $\langle\rangle$. Given $k < \omega$ and a sequence s of length at least k, $s|k$ is the prefix of s of length k. Given a non-empty set X, the set of all finite sequences of elements of X is denoted by $Seq(X)$ and the set of all infinite sequences of elements of X by X^ω. A *tree* on X is a set R of finite sequences of members of X such that if $u \in R$ and v is a prefix of u, then $v \in R$.

A PI-game is based on a non-empty set X, called the *set of moves* that are available to the two players. Intuitively, Player I initially chooses $x_0 \in X$, then Player II chooses $x_1 \in X$, and so on. As it usually happens with everyday games, we would not like the moves made by the two players to be arbitrary. This means that we need a set of *rules* that will impose restrictions on the moves of the two players. The rules are usually (see for example [9]) modeled by a tree R on X: the game must proceed along some branch of R, otherwise the first player who gets outside R looses. The rules of the game will usually be defined by putting down restrictions on the choice of x_n that depend on the preceding moves $x_0, \ldots, x_{n-1}$. The tree R is then obtained in the obvious way:

$\langle x_0, \ldots, x_{n-1}\rangle$ is a path in $R \Leftrightarrow$ for each $i < n$, x_i is allowed by the restrictions

Based on the set X, we define two sets A and B such that A is the set of strategies for Player I and B the set of strategies for Player II. A strategy $a \in A$ assigns a move to each even length partial play of the game; similarly for $b \in B$ and odd length partial plays. Additionally, we assume the existence of a set D, called the set of rewards, which models the potential profit that

a player will have after winning the game. Finally, we consider a function Φ, called the payoff function, which calculates the reward that the winner of a play of the game will get. Usually, the definition of Φ depends on the set of rules R in the sense that if during a play one of the players first breaks the rules, then this should be reflected by the value that Φ returns for that play. The above notions are formalized as follows:

Definition 14 *An infinite game of perfect information (or a PI-game for short) is a sextuple* $\Gamma = (X, R, A, B, D, \Phi)$ *such that:*

- X *is a non-empty set, called the set of* moves *for Players I and II.*
- R *is a* tree *on* X *(usually implicitly specified by a set of rules) which imposes restrictions on the moves of the two players.*
- A *is the* set of strategies for Player I*, which consists of all functions* $a : \bigcup_{n<\omega} X^{2n} \to X$*, with* $X^0 = \{\langle\rangle\}$*.*
- B *is the* set of strategies for Player II*, which consists of all functions* $b : \bigcup_{n<\omega} X^{2n+1} \to X$*.*
- D *is a linearly ordered set called the* set of rewards*, with the property that for all* $S \subseteq D$*,* $lub(S)$ *and* $glb(S)$ *belong to* D*.*
- $\Phi : X^\omega \to D$ *is the* payoff-function *of the game.*

Games of the above form will often be referred as games with payoff*.*

We now define the notion of a *play* of the game:

Definition 15 *Let* $\Gamma = (X, R, A, B, D, \Phi)$ *be a game and let* $a \in A$ *and* $b \in B$ *be two strategies. We define the following sequence:*

$$\begin{aligned} s_0 &= a(\langle\rangle) \\ s_{2i} &= a(\langle s_0, \ldots, s_{2i-1}\rangle) \\ s_{2i+1} &= b(\langle s_0, \ldots, s_{2i}\rangle) \end{aligned}$$

A (complete) play *of the game determined by the strategies* a *and* b *is the infinite sequence* $\langle s_0, s_1, s_2, \ldots\rangle$*. The* s_i*'s will be called the* moves of the play*. A prefix of a play is called a* partial play*.*

Given two strategies $a \in A$ and $b \in B$, we will often write $a \star b$ for the play determined by these two strategies. Given a play s, we will say that a player *first breaks the rules in* s if the first move in s that does not conform to the rules of the game is played by that particular player. A play s will be called *legal* if all its moves conform to the rules of the game.

A special case of games which has a great deal of interest for descriptive set

theory [9], is that of the so-called win-lose games:

Definition 16 *Let* $\Gamma = (X, R, A, B, \{0,1\}, \Phi)$ *be a game and let* $S \subseteq X^\omega$ *be a set. Assume that the payoff function* Φ *of* Γ *is defined as follows:*

$$\Phi(s) = \begin{cases} 1, \textit{ (if Player II first breaks the rules } R \textit{ in } s) \textit{ or } (s \in S) \\ 0, \textit{ otherwise} \end{cases}$$

Then, Γ *will be called a* win-lose game on S.

A special case of the above definition arises when $R = Seq(X)$, ie., when there are no restrictions on the moves of the two players.

A notion that plays a very important role in the theory of infinite games is that of *determinacy* (which is often used in conjunction with the notion of the *value* of a game):

Definition 17 *A game* $\Gamma = (X, R, A, B, D, \Phi)$ *is* determined *with* value v *if*

$$glb_{b\in B}\, lub_{a\in A}\, \Phi(a \star b) \;=\; lub_{a\in A}\, glb_{b\in B}\, \Phi(a \star b) \;=\; v$$

The following inequality (see for example [10]) holds for all games, determined or not:

Proposition 18 *Let* $\Gamma = (X, R, A, B, D, \Phi)$ *be a game (determined or not). Then:*

$$glb_{b\in B}\, lub_{a\in A}\, \Phi(a \star b) \;\geq\; lub_{a\in A}\, glb_{b\in B}\, \Phi(a \star b)$$

The notion of *optimal strategies* is closely connected to determinacy:

Definition 19 *Let* $\Gamma = (X, R, A, B, D, \Phi)$ *be a game with value* v. *The set of* optimal strategies *for Player I (respectively Player II), is denoted by* Opt^I_Γ *(respectively* Opt^{II}_Γ*) and defined as:*

$$Opt^I_\Gamma \;= \{a \in A \mid \forall b \in B,\ \Phi(a \star b) \geq v\}$$
$$Opt^{II}_\Gamma \;= \{b \in B \mid \forall a \in A,\ \Phi(a \star b) \leq v\}$$

We will often simply write Opt^I (respectively Opt^{II}) when the corresponding game is obvious from context.

Notice that it is possible for a game to be determined but no optimal strategies need to exist (see for example [10]). However, as we will later discuss, for the games we consider optimal strategies do exist.

The determinacy of win-lose games has been widely studied and many important results have been obtained. In particular, if Γ is a win-lose game on a set S, then the properties of S play a very important role on the determinacy of Γ. The following two definitions present the classes of open and Borel sets for which determinacy is guaranteed.

Definition 20 *Let X be a non-empty set and let $S \subseteq X^\omega$. Then, S is called* open *if for all $s \in X^\omega$ it is:*

$$s \in S \Rightarrow \exists t \in \omega(\forall s' \in X^\omega[(s|t = s'|t) \Rightarrow s' \in S])$$

In other words, a set S is called open if for any s in S the fact that s is a member of S can be deduced from some finite amount of knowledge about s. For example, take as X the set of natural numbers; then, the set of all sequences with some occurrence of 0 is open, whereas the set of increasing sequences is not.

The *finite Borel hierarchy* is generated by closing out the class of open sets under the operations of union and complementation (relative to X^ω). More formally:

Definition 21 *Let X be a non-empty set. Given $S \subseteq X^\omega$, let $\overline{S}$ denote the complement of S relative to X^ω. Then, the* finite Borel hierarchy *with respect to X is defined as follows:*

- Σ^0_1 *is the collection of all open sets of elements of X^ω*
- *For all $n \geq 1$, $\Pi^0_n = \{\overline{S} \mid S \in \Sigma^0_n\}$*
- *For all $n \geq 1$, $\Sigma^0_{n+1} = \{\bigcup_{i \in \omega} S_i \mid S_i \in \Pi^0_n\}$.*

We then have the following important theorem due to D. Martin [8]:

Theorem 22 (Borel Determinacy Theorem) *Let X be a non-empty countable set and let $S \subseteq X^\omega$. Let Γ be a win-lose game on S. If S is Borel then Γ is determined.*

Despite the fact that the games we will consider in this paper are not strictly win-lose, the above theorem will play an important role in establishing their determinacy.

5 A Formal Definition of the Negation Game

In this section we give a precise definition of the game for logic programs with negation. Let P be a logic program and G a goal clause. We define a PI-game

$\Gamma_{P_G} = (X, R, A, B, D, \Phi)$, which we will often call a *negation game*, as follows:

5.1 The set of moves

The set of moves X of Γ_{P_G} is equal to:

$$X = \{G\} \cup P \cup \mathit{literals}(P_G) \cup \mathit{negvars}(P)$$

In other words, a player can choose one of the following moves: a) he can play the goal clause, or b) play a clause of the program, or c) a literal that appears in G or in the body of a clause of P, or finally, d) an atom that appears in a negative literal in the body of some clause of p.

5.2 The rules of the game

We can now specify the rules that the two players must obey:

- (R1) The first move of Player I is the goal clause G.
- (R2) If the previous move is a clause, the next move is one of the literals in the body of the clause.
- (R3) If the previous move is a positive literal p, the next move is a clause in P whose head is p.
- (R4) If the previous rule is a negative literal $\sim p$, the next move must be p itself (this last move is called a *role-switch*).

Notice that if in rule (R2) the body of the clause is empty, then we will say that the player is *forced to break rule (R2)*. Similarly, the player *is forced to break rule (R3)* if he can not can find a clause in P whose head is p. If one of the players breaks the rules without being forced to, we will say that he *breaks the rules without reason*. This last case refers to moves that are completely unreasonable (such as for example if Player I does not play the goal clause as his first move, or if a player does not choose a literal from the non-empty body of the clause that the other player has just played, etc). We should note here that since our game is infinite, a play continues even after one of the two players has broken the rules; however, the moves beyond this point will be irrelevant to the outcome of the play.

5.3 The sets of strategies

The sets of strategies for the game are specified as in Definition 14 (page 12).

5.4 The set of rewards

The set D of rewards is the set $\{F, 0, T\}$. Intuitively, F corresponds to the *False* truth value, T to the *True* truth value and 0 to an intermediate truth value that is above *False* and below *True*. From the game point of view, F corresponds to a win of Player II, T to a win of Player I, and 0 to a tie of the two Players.

5.5 The payoff function

Let $a \in A$ and $b \in B$ be two strategies, and let $s = a \star b$ be the unique play determined by a and b. Before defining the payoff function, we need to specify when a play can be characterized as a win for Player I (respectively, Player II), ie., when it is a *true-play* (respectively, *false-play*). The conditions used by the two following definitions, will be explained just after the definitions.

Definition 23 *Let P be a program, G a goal, and let s be a play of the corresponding game Γ_{P_G}. Then, s is called a* true-play *if either Player II first breaks the rules in s or if s is a legal play that contains an odd number of negative literals.*

Definition 24 *Let P be a program, G a goal, and let s be a play of the corresponding game Γ_{P_G}. Then, s is called a* false-play *if either Player I first breaks the rules in s or if s is a legal play that contains an even number of negative literals.*

Intuitively, a true-play (respectively, false-play) is a play that is won by Player I (respectively, Player II). More specifically, a play s can be characterized as a true-play (respectively, false-play) if Player II (respectively, Player I) first breaks the rules in s. On the other hand, if s is legal (ie., it has no rule violations) and it contains a finite number of role-switches, then one of the two players remains a doubter after the last role-switch; we would like this particular player to be the winner of the play. So, we keep track of which player remains the doubter, by counting the number of negative literals (or role-switches) in the play. In the beginning of the play, it is Player II who is the doubter, and therefore an even number of role-switches means that Player I will be the doubter after the role-switches end. Similarly, when an even number of role-switches takes place, it will be Player I who will remain a doubter at the end.

We are now in a position to give a formal definition of the payoff function Φ:

$$\Phi(s) = \begin{cases} T, \text{ if } s \text{ is a true-play} \\ F, \text{ if } s \text{ is a false-play} \\ 0, \text{ otherwise} \end{cases}$$

Notice that in the above definition of the payoff function, the value 0 corresponds to the case where there is an infinite number of role switches in the play.

5.6 Certain Examples

We now illustrate the above definitions with certain examples:

Example 25 *Consider the program P:*

$p \leftarrow \sim q, r$

$q \leftarrow \sim s$

$r \leftarrow r$

$s \leftarrow$

and the goal $G = \leftarrow p$. Consider a play of the following form:

Player I	Player II
$\leftarrow p$	p
$p \leftarrow \sim q, r$	$\sim q$
q	$q \leftarrow \sim s$
$\sim s$	s
$s \leftarrow$	$\cdots$
$\cdots$	$\cdots$

In a play of this form, it is Player II who first breaks the rules, and therefore it is a true-play. Therefore, the payoff is equal to T. Another play of this game

is the following:

Player I	Player II
$\leftarrow p$	p
$p \leftarrow\sim q, r$	r
$r \leftarrow r$	r
$\ldots$	$\ldots$

This is obviously a legal play (ie., no player ever breaks the rules) and it is a false-play since it contains an even number of negative literals (zero). Consequently, the payoff in this case is equal to F.

Example 26 *Consider the program P:*

$$p \leftarrow \sim p$$

and the goal $G = \leftarrow p$. Then, there is a unique legal play of the game, namely:

Player I	Player II
$\leftarrow p$	p
$p \leftarrow\sim p$	$\sim p$
p	$p \leftarrow\sim p$
$\sim p$	p
$\vdots$	$\vdots$

It is easy to see that this play is neither a true-play nor a false-play and therefore the payoff in this case is equal to 0.

6 Determinacy of the Negation Game

In this section we demonstrate that given a program P and a goal $\leftarrow p$, the game $\Gamma_{P\cup\{\leftarrow p\}}$ is determined, ie., it has a value. In other words, there exists a value (either F, 0 or T) that is associated with this game (and therefore with p). Since for every atom of P there exists a corresponding game that has a value, this implies that to every atom of the program there corresponds a distinguished value. All these atom-value pairs constitute a three-valued interpretation of P which as we will demonstrate is actually a model of P

(and which as we are going to see later-on coincides with the well-founded model of P).

A basic difference between the negation game and the games that are customarily used in descriptive set theory is that the former is a three-valued one (and not just win-lose), and therefore it is not immediately obvious that it is determined. However, as we are going to demonstrate, we can adapt the powerful results that hold for win-lose games in order to apply to our case. The following lemma provides a sufficient condition for a three-valued game to be determined. Notice that the result holds for any three-valued game that satisfies the requirements of the lemma (and not just for the negation game). This result (which we have not seen explicitly stated elsewhere) was inspired by Corollary 3.3 of [10] that applies to games that have real-valued payoff functions.

Lemma 27 *Let $\Gamma = (X, R, A, B, \{F, 0, T\}, \Phi)$ be a negation game. Assume that Φ has the property that for every $v \in \{F, 0, T\}$ the set $S(v) = \{s \in X^\omega \mid \Phi(s) \geq v\}$ is Borel. Then, Γ is determined.*

PROOF. The basic idea of the proof is to use the Borel determinacy result for win-lose games in order to derive the determinacy of this more complicated game. More specifically, for every $v \in \{F, 0, T\}$, we define the win-lose game $\Gamma_v = (X, R', A, B, \{0, 1\}, \Phi_v)$ on the set $S(v)$, where $R' = Seq(X)$ (ie., there are no restrictions for the players of the game). By the Borel Determinacy Theorem, every Γ_v is determined. Let $v^* = max\{v \in \{F, 0, T\} \mid Opt^I_{\Gamma_v} \neq \emptyset\}$ ie., v^* is the maximum of all v such that the corresponding win-lose game Γ_v is a win for Player I. Notice that v^* is well-defined because there exists at least one of the Γ_v's that is a win for Player I: the game Γ_F (for which $S(F) = X^\omega$ and therefore Player I wins in every case). We distinguish three cases:

Case 1: $v^* = T$. Then, by the definition of v^*, $Opt^I_{\Gamma_{v^*}}$ is not empty. Choose $a_0 \in Opt^I_{\Gamma_{v^*}}$. Then, for any $b \in B$, we have that $a_0 \star b \in S(v^*)$, since a_0 is a winning strategy for Player I in game Γ_{v^*}. But then, by the definition of $S(v^*)$, we have that $\Phi(a_0 \star b) \geq v^*$. Therefore, we have:

$$lub_{a \in A}\, glb_{b \in B}\, \Phi(a \star b) \;\geq\; glb_{b \in B}\, \Phi(a_0 \star b) \;\geq\; v^* = T$$

The above together with Proposition 18 immediately imply that in this case the game Γ_{P_G} is determined with value $v^* = T$.

Case 2: $v^* = 0$. Using exactly the same reasoning as in Case 1 we get that:

$$lub_{a \in A}\, glb_{b \in B}\, \Phi(a \star b) \;\geq\; 0$$

Take now $v = T$. Then, $Opt^{II}_{\Gamma_v}$ is not empty because $v^* = max\{v \in \{F, 0, T\} \mid Opt^{I}_{\Gamma_v} \neq \emptyset\}$ and because the game Γ_v is determined. Then, there exists $b_0 \in Opt^{II}_{\Gamma_v}$ such that for any $a \in A$, $a \star b_0 \notin S(v)$. By the definition of $S(v)$, we have that $\Phi(a \star b_0) < v = T$. Therefore:

$$glb_{b\in B}\, lub_{a\in A}\, \Phi(a \star b) \;\leq\; lub_{a\in A}\, \Phi(a \star b_0) \;<\; T$$

Using the above two facts together with Proposition 18, we get that in this case the game Γ_{P_G} is determined with value $v^* = 0$.

Case 3: $v^* = F$. Take $v = 0$. As in Case 2, we get that there exists $b_0 \in Opt^{II}_{\Gamma_v}$ such that for any $a \in A$, $\Phi(a \star b_0) < v$. This gives:

$$glb_{b\in B}\, lub_{a\in A}\, \Phi(a \star b) \;\leq\; lub_{a\in A}\, \Phi(a \star b_0) \;<\; v \;=\; 0$$

This immediately implies that:

$$glb_{b\in B}\, lub_{a\in A}\, \Phi(a \star b) \;=\; F$$

Using this fact together with Proposition 18, we get that in this case the game Γ_{P_G} is determined with value $v^* = F$. This completes the proof of the proposition. □

The above proposition suggests that in order to demonstrate that the negation game is determined, it suffices to establish that the sets $S(F)$, $S(0)$ and $S(T)$ are Borel. The definition and the two propositions that follow (see for example [9] for the corresponding proofs) provide a convenient tool for establishing that a given set is Borel:

Definition 28 *Let X be a non-empty set and let $G \subseteq X^\omega \times \omega^k$, $k \in \omega$. Then, G will be called* open *if for all $s \in X^\omega$ and for all $n_1, \ldots, n_k \in \omega$ it is:*

$$(s, n_1, \ldots, n_k) \in G \Rightarrow \exists t \in \omega(\forall s' \in X^\omega[(s|t = s'|t) \Rightarrow (s', n_1, \ldots, n_k) \in G])$$

The complement of an open set is a closed *set.*

Proposition 29 *Let X be a set and S be a subset of X^ω. Then, if n is odd, S is Σ^0_n if and only if there exists an open set $G \subseteq X^\omega \times \omega^{n-1}$, such that:*

$$s \in S \Leftrightarrow \exists t_1 \forall t_2 \exists t_3 \cdots \forall t_{n-1} G(s, t_1, \ldots, t_{n-1})$$

Similarly, if n is even, then S is Σ^0_n if and only if there is a closed F such that

$$s \in S \Leftrightarrow \exists t_1 \forall t_2 \exists t_3 \cdots \exists t_{n-1} F(s, t_1, \ldots, t_{n-1})$$

Proposition 30 *Let X be a set and S be a subset of X^ω. Then, if n is odd, S is Π^0_n if and only if there exists a closed set $F \subseteq X^\omega \times \omega^{n-1}$, such that:*

$$s \in S \Leftrightarrow \forall t_1 \exists t_2 \forall t_3 \cdots \exists t_{n-1} F(s, t_1, \ldots, t_{n-1})$$

Similarly, if n is even, then S is Π^0_n if and only if there is an open set G such that

$$s \in S \Leftrightarrow \forall t_1 \exists t_2 \forall t_3 \cdots \forall t_{n-1} G(s, t_1, \ldots, t_{n-1})$$

Since an open (respectively closed) set defines a relation in the obvious way, we will often use the term *open relation* (respectively *closed relation*).

It therefore now remains to show that the sets $S(F)$, $S(0)$ and $S(T)$ of Lemma 27 are Borel. This is demonstrated by the following lemma:

Lemma 31 *The sets $S(F)$, $S(0)$ and $S(T)$ are Borel.*

PROOF. The set $S(F) = \{s \in X^\omega \mid \Phi(s) \geq F\}$ is trivially Borel since it coincides with X^ω. Consider now the case $S(T) = \{s \in X^\omega \mid \Phi(s) = T\}$. Recall now (Definition 23) that $\Phi(s) = T$ iff s is a true-play, ie., iff either Player II first breaks the rules in s or s is a legal sequence that contains an odd number of negative literals. It suffices to express this condition in the notation of one of the above propositions. We express $S(T)$ as:

$$S(T) = \{s \mid \forall t_1 \exists t_2 \forall t_3 \exists t_4 \forall t_5 G(s, t_1, t_2, t_3, t_4, t_5)\}$$

The relation G is defined as:

$$G(s, t_1, t_2, t_3, t_4, t_5) = G_1(s, t_1, t_2, t_3) \vee G_2(s, t_4)$$

where G_1 expresses the condition "is a legal sequence that contains an odd number of negative literals" and G_2 the condition "Player II first breaks the rules". Notice that in the definition of G, t_5 is vacuously quantified (it does not appear in the right hand side of the definition). The definitions of G_1 and G_2 have as follows:

$$\begin{aligned} G_1(s, t_1, t_2, t_3) = [&(\langle s_0, \ldots, s_{t_1 - 1}\rangle \text{ is a legal partial play}) \\ &\wedge (\langle s_0, \ldots, s_{t_2 - 1}\rangle \text{ has an odd number of negative literals}) \\ &\wedge ((t_3 > t_2) \Rightarrow (s_{t_3} \text{ is not a negative literal}))] \end{aligned}$$

and:

$$\begin{aligned} G_2(s, t_4) = [&(\langle s_0, \ldots, s_{2t_4}\rangle \text{ is a legal partial play}) \\ &\wedge (\langle s_0, \ldots, s_{2t_4 + 1}\rangle \text{ is not a legal partial play})] \end{aligned}$$

It is easy to see that G is open: take $t = max\{t_1 - 1, t_2 - 1, t_3, 2t_4 + 1\}$ in the definition of open set (Definition 28).

The case for $S(0)$ is similar: it suffices to express the condition "s is a legal play that has an infinite number of negative literals or s is a true-play". We omit the details. □

We therefore have the following corollary that results directly from the two lemmas of this section:

Corollary 32 *Let P be a program, G a goal clause and let Γ_{P_G} be the corresponding game. Then, Γ_{P_G} is determined.*

Since the negation game is determined, we have the following definition:

Definition 33 *Let P be a program. We define the* game interpretation *N_P of P as the interpretation such that for every $p \in B_P$, $N_P(p)$ is equal to the value of the game $\Gamma_{P\cup\{\leftarrow p\}}$.*

The following theorem states that the game interpretation of a program is actually a model of the program:

Theorem 34 *Let P be a program. Then, N_P is a model of P.*

PROOF. Assume N_P is not a model of P. Then, there must exist a clause $p \leftarrow l_1, \ldots, l_n$ such that $N_P(p) < N_P(l_1, \ldots, l_n)$, ie., $N_P(p) < N_P(l_i)$, for all $i \leq n$. Consider now a strategy a for Player I which chooses as his second move the above clause. Now, if Player II chooses a positive literal l_i from this clause, then the strategy a proceeds in the same way as an optimal strategy of Player I for the game $\Gamma_{P\cup\{\leftarrow l_i\}}$. If on the other hand Player II chooses a negative literal $l_i = \sim q_i$, then the strategy a proceeds according to an optimal strategy of Player II for the game $\Gamma_{P\cup\{\leftarrow q_i\}}$. Now, since $N_P(p) < N_P(l_i)$ for all $i \leq n$, the strategy a when played against any strategy b of Player II, gives a payoff which is greater than $N_P(p)$. Therefore, the value of the game $\Gamma_{P\cup\{\leftarrow p\}}$ is greater than $N_P(p)$ (contradiction). Consequently, N_P is a model of P. □

The above theorem provides a novel, purely game-theoretic characterization of the semantics of negation in logic programming. In the next sections we investigate how this approach relates to the existing semantic approaches for negation.

7 The Refined Negation Game

In the next section we will demonstrate the equivalence of the game semantics to the well-founded model of negation in logic programming. As we have already mentioned, the well-founded model is usually constructed in stages. For this reason, in this paper we have adopted the infinite-valued characterization of the well-founded model which makes these stages more explicit. On the other hand, the three-valued negation game that we have defined does not have any notion of "stages". This difference between the well-founded model (or the infinite-valued model) and the three-valued game makes it difficult at first sight to establish their equivalence. The solution we have adopted is to refine the negation game getting a new game in which the two players can have degrees of winning and losing. These degrees correspond to the stages of the well-founded model (and correspondingly to the different levels of truth values of the infinite-valued model). The existence of "stages" in our formalisms allows us to prove results by induction. Therefore, in this section we will define the refined negation game and we will demonstrate that it is determined.

The basic idea behind the refined game is that the payoff of a given play will depend on the number of role-switches that have taken place. More specifically, assume that we have a legal play or a play in which the rules are first broken because one of the players is forced to break them. Then, if the play is a true-play (respectively false-play), the payoff is going to be T_k (respectively F_k), where k is the number of role switches that have taken place. If on the other hand we have a play in which the rules are first broken without reason by Player I (respectively Player II) then the other player gets the maximum possible payoff, namely F_0 (respectively T_0).

More formally now, we define $\hat{s}$ to be the maximum initial segment of s in which the rules have not been broken; in particular, $\hat{s} = s$ if s is a legal play. Moreover, we define $\parallel s \parallel$ as follows: if s is not a legal play and the first violation of the rules in s is without reason, then $\parallel s \parallel = 0$; otherwise, $\parallel s \parallel$ is equal to the number of negative literals in $\hat{s}$.

We can now give the formal definition of the refined game. We need to refine the definition of the set of rewards and of the payoff function, as follows:

The set of rewards: The set D of rewards is the set $\{F_0, F_1, \ldots, 0, \ldots, T_1, T_0\}$ of truth values which are ordered as: $F_0 < F_1 < \cdots < 0 < \cdots < T_1 < T_0$.

The payoff function: The refined payoff function Φ is defined as follows:

$$\Phi(s) = \begin{cases} T_{\|s\|}, \text{ if } s \text{ is a true-play} \\ F_{\|s\|}, \text{ if } s \text{ is a false-play} \\ 0, \quad \text{otherwise} \end{cases}$$

We now argue about the determinacy of this refined game. The proof is similar in spirit but more complicated than that of Lemma 27:

Lemma 35 *Let $\Gamma = (X, R, A, B, V, \Phi)$ be a refined negation game. Assume now that Φ has the property that for every $v \in V$ the set $S(v) = \{s \in X^\omega \mid \Phi(s) \geq v\}$ is Borel. Then, Γ is determined.*

PROOF. The basic idea is to use the Borel determinacy of simple-win lose games in order to establish the determinacy of the refined game. More specifically, for every $v \in V$, we define the win-lose game $\Gamma_v = (X, R', A, B, \{0,1\}, \Phi_v)$ on the set $S(v)$, where $R' = Seq(X)$. By the Borel Determinacy Theorem, every Γ_v is determined. Let $v^* = lub\{v \in V \mid Opt^I_{\Gamma_v} \neq \emptyset\}$ ie., v^* is the least upper bound of all v such that the corresponding win-lose game Γ_v is a win for Player I. Notice that v^* is well-defined because there exists at least one of the Γ_v's that is a win for Player I: the game Γ_{F_0} (for which $S(F_0) = X^\omega$ and therefore Player I wins in every case).

Case 1: $v^* = F_k$, $k \geq 0$. By the definition of v^*, $Opt^I_{\Gamma_{v^*}}$ is not empty. Choose $a_0 \in Opt^I_{\Gamma_{v^*}}$. Then, for any $b \in B$, we have that $a_0 \star b \in S(v^*)$, since a_0 is a winning strategy for Player I in game Γ_{v^*}. But then, by the definition of $S(v^*)$, we have that $\Phi(a_0 \star b) \geq v^*$. Therefore:

$$lub_{a \in A}\, glb_{b \in B}\, \Phi(a \star b) \;\geq\; glb_{b \in B}\, \Phi(a_0 \star b) \;\geq\; v^* = F_k$$

Let now $v = F_{k+1}$. Then, $Opt^{II}_{\Gamma_v}$ is not empty because $v^* = lub\{v \in V \mid Opt^I_{\Gamma_v} \neq \emptyset\}$ and because the game Γ_v is determined. Then, there exists $b_0 \in Opt^{II}_{\Gamma_v}$ such that for any $a \in A$, $a \star b_0 \notin S(v)$. By the definition of $S(v)$, we have that $\Phi(a \star b_0) < v$. Therefore:

$$glb_{b \in B}\, lub_{a \in A}\, \Phi(a \star b) \;\leq\; lub_{a \in A}\, \Phi(a \star b_0) \;<\; v = F_{k+1}$$

Combining the two inequalities that we have obtained together with Proposition 18, we get the desired result for the case $v^* = F_k$.

Case 2: $v^* = 0$. Then, exactly as in Case 1, it can be shown that:

$$lub_{a \in A}\, glb_{b \in B}\, \Phi(a \star b) \;\geq\; v^*$$

Consider now on the other hand any v such that $v > v^*$. Then, $Opt^{II}_{\Gamma_v}$ is not empty because $v^* = lub\{v \in V \mid Opt^{I}_{\Gamma_v} \neq \emptyset\}$ and because the game Γ_v is determined. Then, there exists $b_0 \in Opt^{II}_{\Gamma_v}$ such that for any $a \in A$, $a \star b_0 \notin S(v)$. By the definition of $S(v)$, we have that $\Phi(a \star b_0) < v$. Therefore:

$$glb_{b \in B}\, lub_{a \in A}\, \Phi(a \star b) \;\leq\; lub_{a \in A}\, \Phi(a \star b_0) \;<\; v$$

Now, since the above holds for all $v > v^*$, we get:

$$glb_{b \in B}\, lub_{a \in A}\, \Phi(a \star b) \;\leq\; v^*$$

Combining the two inequalities that we have obtained together with Proposition 18, we get the desired result for the case $v^* = 0$.

Case 3: $v^* = T_k$, $k \geq 0$. For the case $v^* = T_0$, it is immediately obvious that:

$$glb_{b \in B}\, lub_{a \in A}\, \Phi(a \star b) \;\leq\; v^*$$

For the case $v^* = T_k$, $k > 0$, take $v = T_{k-1}$. The proof then follows in the same way as in Case 1. □

It now remains to demonstrate that each of the sets $S(v)$ of the above proposition, is Borel:

Lemma 36 *The sets $S(v)$, $v \in V$, are Borel.*

PROOF. Similar to the proof of Lemma 31. □

We therefore have the following corollary that follows directly from the two lemmas of this section:

Corollary 37 *Let P be a program, G a goal clause and let Γ_{P_G} be the corresponding refined negation game. Then, Γ_{P_G} is determined.*

8 Equivalence of the Game to the Well-Founded Model

Consider a given logic program P. What we would like to demonstrate is that the value that the infinite-valued model assigns to an atom p, coincides with

the value of the game $\Gamma_{P\cup\{\leftarrow p\}}$. This can be demonstrated by simultaneously showing (by induction on k) the following two statements:

- If the value of the game $\Gamma_{P\cup\{\leftarrow p\}}$ is T_k (or F_k), then the value assigned to p by the minimum infinite-valued model is T_k (respectively F_k).
- If the value assigned to p by the minimum infinite-valued model is T_k (or F_k) then the value of the game $\Gamma_{P\cup\{\leftarrow p\}}$ is T_k (respectively F_k).

The first of the above two statements is easier to establish: we define a quantity called the *depth* of the game and then show the desired statement by an inner induction on the depth (actually, the depth is only needed for the T_k case).

The second statement is trickier to demonstrate. We give an intuitive description of the approach we use. Assume that the minimum infinite-valued model assigns to p the value T_k (similar arguments apply for F_k). Then, we must show that there exists a strategy for Player I such that for all strategies of Player II, the payoff is greater than or equal to T_k. In other words, the desired strategy for Player I must not be arbitrary: it must make careful selections in order to corner Player II to a value greater than or equal to T_k. Now, since the value that the model assigns to p is equal to T_k, there must exist a clause in P whose body evaluates (under the infinite-valued model) to T_k. This is a good choice for Player I because no-matter which literal Player II will choose from the body of the clause, Player I can ensure a payoff of at least T_k. But then, using the induction hypothesis and the determinacy of the refined game, it can be shown that the best possible situation for the two players occurs when the payoff is exactly T_k. This intuitively establishes the desired result.

We can now proceed with the formal details of the proof. We start with the following simple lemma:

Lemma 38 *Let Γ_{P_G} be a refined negation game with value $v \neq 0$. Then, the two players have optimal strategies (ie., the sets Opt^I and Opt^{II} are non-empty).*

PROOF. Since the value of the game is either T_k or F_k for some $k \in \omega$, and since there are no arbitrarily close approximations to T_k and F_k other than themselves, the result follows immediately. □

We now define a quantity that will be used in the definition of depth:

Definition 39 *Let $\Gamma_{P_G} = (X, R, A, B, V, \Phi)$ be a refined negation game with value $v > 0$ and let s be the play determined by two strategies $a \in Opt^I_{\Gamma_{P_G}}$ and $b \in Opt^{II}_{\Gamma_{P_G}}$. Then, by $\ll s \gg$ we denote the number of moves that Player II has played in s before either a fact or a negative literal is encountered.*

We can now define the "depth" of a given game. Assume we consider a game Γ_{P_G} that has value $v > 0$. Then, by Lemma 38, the sets $Opt^I_{\Gamma_{P_G}}$ and $Opt^{II}_{\Gamma_{P_G}}$ of optimal strategies of the game are non-empty. We have:

Definition 40 *Let $\Gamma_{P_G} = (X, R, A, B, V, \Phi)$ be a refined negation game with value $v > 0$. Define:*

$$d = min_{a \in Opt^I}\, max_{b \in Opt^{II}}\ \ll a \star b \gg$$

The quantity d is called the depth *of the game Γ_{P_G}.*

Intuitively, the depth expresses the maximum number of moves that Player II can play before a negative literal or a fact is encountered, in the case that Player I plays in the best possible way. It is not hard to see that the notion of depth is well-defined, ie., that it is always finite. If Player II could (against any strategy of Player I) make the game last arbitrarily long before a negative literal or a fact is encountered, then (by König's Lemma) he would be able to make it last for ever. But in this case the value of the game would be F_0, which contradicts our assumption in Definition 40 that the value of the game is greater than 0.

We can now establish the equivalence of the refined game with the infinite-valued semantics:

Theorem 41 *Let P be a program and let p be an atom that appears in P. Consider the goal $G = \leftarrow p$ and let $\Gamma_{P_G} = (X, R, A, B, V, \Phi)$ be the corresponding refined game. Moreover, let M_P be the minimum infinite-valued model of P. Then, Γ_{P_G} has value T_k (respectively F_k) if and only if $M_P(p) = T_k$ (respectively $M_P(p) = F_k$).*

PROOF. The proof is by induction on k.

<u>*Basis Case:*</u> We demonstrate the case $k = 0$.

($\Rightarrow$) Assume that the game has value T_0 (the proof for F_0 is actually easier since it does not use the notion of depth, and therefore omitted). We demonstrate the desired result by induction on the depth of the game. In particular we show that for all p in P, if $\Gamma_{P \cup \{\leftarrow p\}}$ has value T_0 and depth d, then $M_P(p) = T_0$. The basis case is for $d = 1$. This can only happen if Player I plays a fact $p \leftarrow$ as his second move (his first move is the goal clause), in which case the desired result is immediate. Assume the result holds for depths $\leq d$. We demonstrate the case $d + 1$. Consider strategies $a \in Opt^I$ and $b \in Opt^{II}$ such that $\ll a \star b \gg = d+1$. Let $p \leftarrow l_1, \ldots, l_r$ be the second move of Player I, in the

play $a \star b$. Now, since $a \in Opt^I$, every play determined by a and any strategy of Player II, always has payoff T_0. But then this easily implies that all the l_i are positive literals and that the games $\Gamma_{P \cup \{\leftarrow l_i\}}$ have value T_0; moreover, their depth is at most d. Therefore, by the induction hypothesis, $M_P(l_i) = T_0$ for all $1 \leq i \leq r$, which implies that $M_P(p) = T_0$ (since M_P is a model of P).

($\Leftarrow$) $M_P(p) = T_0$ (the case $M_P(p) = F_0$ is similar and omitted). We demonstrate the following statement using induction: for all p in P and for all $m \in \omega$, if $T_P^m(\emptyset)(p) = T_0$, then $\Gamma_{P \cup \{\leftarrow p\}}$ has value T_0. The basis case is for $m = 0$ and it vacuously holds. Assume the statement holds for m. We demonstrate the case $m + 1$, ie., we show that for all p in P, if $T_P^{m+1}(\emptyset)(p) = T_0$, then $\Gamma_{P \cup \{\leftarrow p\}}$ has value T_0. Obviously, there must exist in P either a fact of the form $p \leftarrow$ or a clause $p \leftarrow l_1, \ldots, l_r$ such that $T_P^m(\emptyset)(l_1, \ldots, l_r) = T_0$. In the latter case, all the l_i must be positive literals and $T_P^m(\emptyset)(l_i) = T_0$ for all i. By the induction hypothesis, the games $\Gamma_{P \cup \{\leftarrow l_i\}}$ all have value T_0. Consider now a strategy $a \in A$ which proceeds as follows: if there exists a fact $p \leftarrow$ in P, then a specifies this fact as the first move of Player I; otherwise, it specifies a clause $p \leftarrow l_1, \ldots, l_r$ as discussed above. Moreover, the subsequent moves of Player I dictated by a ensure a payoff T_0 (we know this is possible because, by the induction hypothesis, the games $\Gamma_{P \cup \{\leftarrow l_i\}}$ all have value T_0). Therefore, the strategy a gives a payoff T_0 against all strategies of Player II; this implies that the game $\Gamma_{P \cup \{\leftarrow p\}}$ has value T_0.

This concludes the proof of $\Leftarrow$ and of the basis case.

Induction Hypothesis: We assume the statement holds for all natural numbers less than or equal to k.

Induction Step: We demonstrate the desired result for $k + 1$.

($\Rightarrow$) We show that if Γ_{P_G} has value T_{k+1} then $M_P(p) = T_{k+1}$ (the case F_{k+1} is similar and omitted). We demonstrate by an inner induction on the depth of the game that for all p in P, if $\Gamma_{P \cup \{\leftarrow p\}}$ has value T_{k+1} and depth d, then $M_P(p) = T_{k+1}$. The basis case is for $d = 1$. The fact that $d = 1$ implies that there exists an optimal strategy of Player I that chooses as the second move of Player I the clause $p \leftarrow E$ such that all literals in E that can be chosen by optimal strategies of Player II, are negative. Let $\sim q$ be one of these literals. Then, by applying the outer induction hypothesis to the game $\Gamma_{P \cup \{\leftarrow q\}}$, one can easily see that $M_P(q) = F_k$ and therefore $M_P(\sim q) = T_{k+1}$. For every other literal l that appears in E it can easily be shown that $M_P(l) \geq T_{k+1}$. Therefore, $M_P(p) \geq T_{k+1}$ (since M_P is a model of P), which (by the outer induction hypothesis) means that $M_P(p) = T_{k+1}$.

Assume now that the result holds for all depths less than or equal to d. We demonstrate the case $d+1$. Consider strategies $a \in Opt^I$ and $b \in Opt^{II}$ such that $\ll a \star b \gg = d+1$. Let $p \leftarrow l_1, \ldots, l_r$ be the second move of Player I under the strategy a and let $p_1, \ldots, p_r$ be the atoms that correspond to the literals $l_1, \ldots, l_r$. Then, it is easy to see that for every one of the l_i one of the following holds:

- $l_i = p_i$ and the game $\Gamma_{P \cup \{\leftarrow p_i\}}$ has value T_{k+1} and depth at most d, or
- $l_i = p_i$ and the game $\Gamma_{P \cup \{\leftarrow p_i\}}$ has value $\geq T_k$, or
- $l_i = \sim p_i$ and the game $\Gamma_{P \cup \{\leftarrow p_i\}}$ has value $\leq F_k$.

In all the above cases, using either the inner or the outer induction hypothesis, it is $M_P(l_i) \geq T_{k+1}$ for all $1 \leq i \leq r$. This implies that $M_P(p) \geq T_{k+1}$ (since M_P is a model of P), which (by the outer induction hypothesis) means that $M_P(p) = T_{k+1}$.

($\Leftarrow$) Assume $M_P(p) = T_{k+1}$ (a similar proof applies in the case where $M_P(p) = F_{k+1}$). Then, there exists m such that $T_P^m(M_k)(p) = T_{k+1}$. We perform an inner induction on m, ie., we demonstrate that if $T_P^m(M_k)(p) = T_{k+1}$ then Γ_{P_G} has value T_{k+1}. The case $m = 0$ vacuously holds because M_k does not assign the value T_{k+1} to any atom. Assume the statement holds for m. We demonstrate the case $m+1$. Assume therefore that $T_P^{m+1}(M_k)(p) = T_{k+1}$. This implies that there exists a clause $p \leftarrow E$ in P such that $T_P^m(M_k)(E) = T_{k+1}$. Assume now that Player I chooses as his second move a clause $p \leftarrow E$ of the above form. Then, no matter how Player II plays, Player I can ensure a payoff of at least T_{k+1}:

- If Player II chooses a positive literal l such that $T_P^m(M_k)(l) \geq T_{k+1}$, then the game $\Gamma_{P \cup \{\leftarrow l\}}$ has (by the outer induction hypothesis or the inner induction hypothesis) value $M_P(l) = T_P^m(M_k)(l) \geq T_{k+1}$.
- If Player II chooses a negative literal $\sim r$, and $T_P^m(M_k)(\sim r) \geq T_{k+1}$, then the game $\Gamma_{P \cup \{\leftarrow r\}}$ has (again by the outer induction hypothesis) value $M_P(r) = T_P^m(M_k)(r) \leq F_k$.

The above discussion suggests that Player I has a winning strategy that ensures a payoff of at least T_{k+1}. Now, since the game is determined (Corollary 37) and the values above T_{k+1} are (by the outer induction hypothesis) reserved for the atoms of P that under M_P have value greater than T_{k+1}, we get that the value of Γ_{P_G} is exactly T_{k+1}. This concludes the proof of the theorem. $\square$

Notice that the above theorem does not state anything about the case where either the game or the minimum infinite-valued model assigns to an atom the value 0. But since the refined game is determined and since the game

and the infinite-valued model agree on all other cases, the following result is immediate:

Theorem 42 *Let P be a program and let p be an atom that appears in P. Consider the goal $G =\leftarrow p$ and let $\Gamma_{P_G} = (X, R, A, B, V, \Phi)$ be the corresponding refined game. Moreover, let M_P be the minimum infinite-valued model of P. Then, Γ_{P_G} has value $v \in V$ if and only if $M_P(p) = v$.*

Therefore, the refined game is equivalent to the minimum infinite-valued model. Now, since the refined game collapses to the unrefined one and the infinite-valued model collapses to the well-founded one, we have:

Theorem 43 *Let P be a program and let p be an atom that appears in P. Consider the goal $G =\leftarrow p$ and let $\Gamma_{P_G} = (X, R, A, B, \{F, 0, T\}, \Phi)$ be the corresponding (unrefined) game. Moreover, let M be the well-founded model of P. Then, Γ_{P_G} has value $v \in \{F, 0, T\}$ if and only if $M(p) = v$.*

This concludes the proof of the equivalence between the well-founded semantics and the simple unrefined negation game that we have introduced.

9 Related and Future Work

We have presented a game semantics for well-founded negation in logic programming. Despite the fact that game semantics are well-established for more mainstream programming languages [2], their application to logic programming has been very restricted. To our knowledge, there exist only two other works that deal with the problem of giving a game semantics to logic programming. However, both of them deal with the negation-free case which is actually simpler. The first of them, appears in [16] in which M. H. van Emden develops a probabilistic version of logic programming whose proof theory is described using a two-person game. This work, although ground-breaking, does not treat negation apart from a short discussion in the concluding section regarding Clark's negation-as-failure (at the time that [16] was published, well-founded negation had not been formalized yet). More recently, the game for the negation-free case was also studied in [4], and interesting connections with the classical semantics of logic programming have been established; however, well-founded negation is still not considered (apart from a short paragraph in the concluding section regarding the possibility of extending the game to programs with negative goals). Less directly connected to our work but very indicative of the rich connections between game theory and logic programming, is the work of M. De Vos (see for example [18], [19], etc). More specifically, in [19] certain new logic programming formalisms are introduced in order to model decision-making. It is demonstrated that strategic games and extensive

games of perfect information can be represented in these new formalisms in such a way that the equilibria of the games can be retrieved as the stable models or answer-sets of the programs.

There are many aspects of this work that we feel that should be further investigated. First of all, we believe that the (unrefined) negation game applies as it is to infinite propositional programs. However, the proof of correctness has to be more involved. This is mainly due to the fact that the construction of the well-founded model of an infinite propositional program may require a transfinite number of iterations. This is also reflected in the construction of the minimum infinite-valued model of such programs: in this case the set V of truth values contains a F_α and a T_α for each countable ordinal α (see [14] for details). Therefore, in the correctness proof for the case of infinite programs, one has to appropriately redefine the refined game so as that the payoff function ranges over this new extended set of truth values. In the theory of infinite games such a situation is usually treated by introducing an auxiliary ordinal in the game that can be considered as a type of clock which imposes a "time limit" to the moves of the players (see for example [20][page 47]). We are currently investigating this issue.

The use of any new semantic approach for a programming language, can only be tested by its applications. We are therefore interested in applying the proposed approach in order to establish properties of logic programs that use well-founded negation. We conjecture that the game semantics can be used to demonstrate the correctness of program transformations as-well-as to define new ones. Since games are intuitive and natural, it is interesting to investigate whether they can offer certain benefits when compared against the classical semantics approaches. Moreover, we would like to extend the game semantics to apply to other logic programming languages. One possible such candidate is disjunctive logic programming (with or without negation in clause bodies).

Acknowledgments: We wish to thank M.H. van Emden for pointing out to us that the game for negation-free programs already existed in his [16]. Special thanks go to the reviewers of the paper for their constructive comments. Finally, the second author would like to thank J. Mycielski for clarifications regarding Corollary 3.3 of [10] and Y. Moschovakis for numerous explanations regarding the theory of infinite games.

References

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

[2] S. Abramsky and G. McCusker. Game Semantics. In H. Schwichtenberg and U. Berger, editor, *Computational Logic: Proceedings of the 1997 Marktoberdorf Summer School*, pages 1–56. Springer-Verlag, 1999.

[3] C. Baral and M. Gelfond. Logic Programming and Knowledge Representation. *Journal of Logic Programming*, 19(20):73–148, 1994.

[4] R. Di Cosmo, J.V. Loddo, and S. Nicolet. A Game Semantics Foundation for Logic Programming. In C. Palamidessi, H. Glaser, and K. Meinke, editor, *Proceedings of PLILP*, volume LNCS 1490, pages 355–373. Springer-Verlag, 1998.

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

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

[7] D. Gale and F.M. Stewart. Infinite Games with Perfect Information. In *Annals of Mathematical Studies*, volume 28, pages 245–266. Princeton University Press, 1953.

[8] D.A. Martin. Borel Determinacy. *Annals of Mathematics*, 102:363–371, 1975.

[9] Y. N. Moschovakis. *Descriptive Set Theory*. Amsterdam: North-Holland, 1980.

[10] J. Mycielski. Games with Perfect Information. In R.J. Aumann and S. Hart, editor, *Handbook of Game Theory*, pages 41–70. Elsevier Science Publishers, 1992.

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

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

[13] P. Rondogiannis and W.W. Wadge. An Infinite-Valued Semantics for Logic Programs with Negation. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, pages 456–467. Springer-Verlag, 2002.

[14] P. Rondogiannis and W.W. Wadge. Minimum Model Semantics for Logic Programs with Negation-as-Failure. *ACM Transactions on Computational Logic*, 6(2):441–467, 2005.

[15] J. Ullman. *Database and Knowledge-Base Systems*. Computer Science Press, 1989.

[16] M.H. van Emden. Quantitative Deduction and its Fixpoint Theory. *Journal of Logic Programming*, 3(1):37–53, 1986.

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

[18] M. De Vos. and D. Vermeir. Choice Logic Programs and Nash Equilibria in Strategic Games. In *Proceedings of the 13th International Workshop on Computer Science Logic (CSL)*, pages 266–276. Springer-Verlag, 1999.

[19] M. De Vos. *Logic Programming, Decisions and Games.* PhD thesis, Vrije Universiteit Brussel, 2001.

[20] W.W. Wadge. *Reducibility and Determinateness on the Baire Space.* PhD Dissertation, University of California, Berkeley, 1984.