Infinite-game Semantics for Logic Programming Languages

Chrysida Galanaki*

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
chrysida@di.uoa.gr

Abstract. This thesis focuses on the study of the semantics of logic programs and the development of infinite games of perfect information between two players, that capture this semantics. We define a game that, given a propositional logic program with negation and a ground atom that belongs to it, may have three possible results (win for any of the players or tie). The game is determined. Based on this, we get a game interpretation of the program that is a model. Moreover, it is equivalent to the well-founded semantics of the program. In order to prove that, we use a new refined game that has infinite possible outcomes. The refined game is also proved to be determined and its game interpretation is a model of the program and equivalent to the infinite valued minimum model semantics of the program. Our study then extends to intensional logic programming a generalization of temporal and modal logic programming. For the monotonic case we develop a game semantics equivalent to the existing semantics. More importantly, we extend it to a three-valued game semantics, the first semantics for the broad class of non-monotonic intensional logic programming languages.

1 Dissertation summary

The purpose of this dissertation is to use infinite games in order to provide an alternative, simple and elegant semantics for logic programming languages. We consider two such languages and provide for each one of them a corresponding infinite game that captures its semantics. The first language is that of normal logic programs, i.e. logic programs that allow negative literals in the bodies of clauses. The second language is intensional logic programming, i.e. logic programming extended with intensional operators.

The semantics of logic programming has been extensively investigated. Probably the most broadly studied topic in the area is the problem of extending logic programming with negation. 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. 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* [35] and

* Dissertation advisor: Panos Rondogiannis, Associate Professor

the *stable model semantics* [14]. The former approach provides a unique "distinguished" model of the program while the latter allows for the possibility of zero, one or many models. This thesis focuses on the well-founded semantics.

Game-theoretical semantics has been extensively studied in logic and language [15]. Despite the fact that game semantics is well-established for more mainstream programming languages [2], their application to logic programming has been very restricted. To our knowledge, there exist 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. The first of them, appears in [34] 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. More recently, the game for the negation-free case was also studied in [6], and interesting connections with the classical semantics of logic programming have been established. Less directly connected to our work but very indicative of the connections between game theory and logic programming, is the work of M. De Vos (see for example [5], [4]). More specifically, in [4] 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.

The starting point of our investigation is the simple game semantics for ordinary negation-less logic programming in [34]. Suppose we have a program P and a goal clause G. 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 variables in G which he thinks will fail on its own, and plays it, thus asserting his doubts. From then on, the play proceeds as follows: the Believer (who thinks the variable just played by the Doubter will in fact succeed) must play a clause in the program whose head is the variable just played; and the Doubter must, on his turn, play one of the variables 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 choses to doubt an atom for which there is a fact in the program. For the Doubter, this means choosing a variable for which there is no rule; in this case the Believer has chosen a rule with a variable in its body for which there is no evidence. Finally, we must give the Doubter an important advantage: he wins if the game never ends.

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 variables which themselves fail. If G succeeds, the Believer's winning strategy is to repeatedly choose rules that are applicable, ie., for which

all the variables in the body succeed. The only subtle point is that the Believer, in choosing applicable rules, must avoid ones like $p \leftarrow p$ 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 formula 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 Player 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 Player 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. Finally, if the players swap roles infinitely often, the result is a tie.

The game we have just described is determined, i.e. it always has a value, and 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.

A characterization of the well-founded model that captures in a logical way this notion of different strengths of truth values has been introduced by P. Rondogiannis and W. W. Wadge in [29, 28]. More specifically, the *infinite-valued semantics* introduced in [29, 28] 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 show that this game is also determined. We then demonstrate that the interpretation of the program that we get using this new infinite-valued game, is equivalent to the infinite-valued semantics. This will immediately imply that the game interpretation of the initial three-valued game is equivalent to the well-founded semantics.

Our study then focuses on *Intensional logic programming*, an extension of logic programming based on intensional logic. *Intensional Logic* is an extension of classical logic that was introduced by R. Montague [17] in order to capture the semantics of natural languages. Roughly speaking, intensional logic was proposed as a formal system for understanding and reasoning about *context-dependent* properties of natural language expressions. In its initial form, intensional logic was a higher-order one, equipped with modal and temporal operators [13]. However, the term "intensional logics" can also be used more loosely in order to describe a large class of logics for reasoning about context-dependent phenomena [20]. *Temporal logics* and *modal logics* are special cases of intensional logic.

Based on this broad interpretation of the term, M. Orgun and W. W. Wadge introduced in [24] the notion of *intensional logic programming*, which includes as special cases many non-classical extensions of logic programming (such as *temporal logic programming*, *modal logic programming*, and so on). As pointed out in [24], numerous logic programming languages that have been proposed in the literature can be characterized as "intensional" (such as Chronolog [24], Tempura [19], Molog [7], Cactus [27], MProlog [22] and so on). It was therefore natural to wonder whether there exists a common semantic framework for handling all these systems in a uniform way. As it was demonstrated in [24], if the intensional operators of the source intensional logic programming language obey some simple semantic properties, then the programs of the language are guaranteed to possess the *minimum model* property. However, all the intensional operators allowed in [24] are assumed to satisfy the *monotonicity* property and this excludes many interesting applications that involve *non-monotonicity*, which is a crucial concept involved in knowledge representation and reasoning.

Our purpose is to extend the framework of [24] to allow arbitrary (non-monotonic) intensional operators in the bodies of program clauses and to define a general semantic framework for non-monotonic intensional logic programming. Our approach is again based on *game semantics*.

We begin by constructing a simple two-person game for the class of intensional logic programs considered in [24] (in which the intensional operators are monotonic, universal and conjunctive). We demonstrate that the outcome of the game coincides with the minimum model semantics obtained in [24]. In this way we provide an equivalent formulation to the approach of Orgun and Wadge for facilitating the further study of intensional logic programs.

We then extend the proposed game to handle intensional logic programs that even use non-monotonic operators in the bodies of clauses and show that the games are determined. In this way we obtain the first general semantic framework for non-monotonic intensional logic programming. It should be noted

that intensional logic programming, due to its variety of operators, allows a much broader framework for non-monotonicity than classical logic programming where the main source of non-monotonicity is the operator of negation-as-failure.

2 Main results

Game Semantics of Normal Logic Programming

Let P be a logic program and G a goal clause. We define a corresponding PI-game $\Gamma_{P_G} = (X, T_\omega, D, \Phi)$, as follows: The set of moves X is:

$X = \{G\} \cup P \cup \mathit{literals}(P_G) \cup \mathit{negvars}(P) \cup \langle \mathit{I've\ lost} \rangle \cup \langle \mathit{I've\ won} \rangle$

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, d) a propositional variable that appears in a negative literal in the body of some clause of p, or finally, declare losing or winning.

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

- (R1) The first move of Player I is the goal clause G and the next move, by Player II, is the literal in the body of the goal clause.
- (R2) If the previous move is a clause (with non-empty body), 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 (and whose body could possibly be empty).
- (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***).
- (R5-6) If none of the above rules is applicable, the player breaks the rules and loses (plays the $\langle \mathit{I've\ lost} \rangle$ move from then on) while the other player wins (plays the $\langle \mathit{I've\ won} \rangle$ move from then on).

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. We should note here that since our game is infinite, a play continues even after one of the two players has broken the rules and the game has essentially ended in favor of one of the two players. The player who is forced to break the rules keeps on playing the move $\langle \mathit{I've\ lost} \rangle$ while the other, who has won the play, keeps on playing the move $\langle \mathit{I've\ won} \rangle$. However, the moves beyond this point will be irrelevant to the outcome of the play. This way every play is infinite. A play that does not contain $\langle \mathit{I've\ won} \rangle$ and $\langle \mathit{I've\ lost} \rangle$ moves will be called a *genuinely infinite play*.

More formally, the infinite tree T_ω of the game Γ_{P_G} consists of all infinite sequences $\langle x_0, x_1, \ldots, x_k, \ldots \rangle$, which satisfy the following restrictions:

$\mathbf{R_1}$: $x_0 = \langle \leftarrow p \rangle$, where $G = \leftarrow p$ is a goal clause, and $x_1 = \langle p \rangle$.
$\mathbf{R_2}$: If $x_k = \langle q \leftarrow q_1, \cdots, q_n \rangle$, then $x_{k+1} = \langle q_i \rangle$, where $0 < i \leq n$.
$\mathbf{R_3}$: If $x_k = \langle p \rangle$, then $x_{k+1} = \langle \mathcal{C} \rangle$, where $\mathcal{C}$ is a clause whose head is p.

$\mathbf{R_4}$: If $x_k = \langle \sim p \rangle$, then $x_{k+1} = \langle p \rangle$.
$\mathbf{R_5}$: If after x_k has been played none of the above rules is applicable, then $x_{k+1} = \langle \textit{I've lost} \rangle$.
$\mathbf{R_6}$: If $x_k = \langle \textit{I've lost} \rangle$, then $x_{k+1} = \langle \textit{I've won} \rangle$ (and vice-versa).

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.

Let $a \in \mathsf{Strat}^I(\Gamma)$ and $b \in \mathsf{Strat}^{II}(\Gamma)$ be two strategies, and let $s = a \star b$ be the unique play determined by a and b. The following two definitions will be useful in defining the payoff function:

Definition 1. *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 plays the $\langle$I've lost$\rangle$ move in s or if s is a genuinely infinite play that contains an odd number of negative literals.*

Definition 2. *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 plays the $\langle$I've lost$\rangle$ move in s or if s is a genuinely infinite play that contains an even number of negative literals.*

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 s is a genuinely infinite play that contains an infinite number of negative literals i.e. there is an infinite number of role switches in the play.

Corollary 1. *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 3. *Let P be a program. We define the* game interpretation *N_P of P as the interpretation such that for every p in the Herbrand base B_P of the program, $N_P(p)$ 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 1. *Let P be a program. Then, 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. Subsequently, we investigate how this approach relates to the existing semantic approaches for negation and we prove the equivalence between N_P and the well-founded model semantics.

Theorem 2. *Let P be a program and let p be an atom that appears in P. Consider the goal $G = \leftarrow p$ and let $\Gamma_{P_G} = (X, T_\omega, \{F, 0, T\}, \Phi)$ 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 $M(p) = v$.*

In order to prove the above theorem we use a new refined game that has infinite possible outcomes. The value of that game depends on the number of role switches that take place during it. The value of the game is T_k (respectively F_k) if Player I (respectively Player II) wins after k role-switches.

Corollary 2. *Let P be a program, G a goal clause and let Γ_{P_G} be the corresponding refined (infinite-valued) negation game. Then, Γ_{P_G} is determined.*

Theorem 3. *Let P be a program and let p be an atom that appears in P. Consider the goal $G = \leftarrow p$ and let $\Gamma_{P_G} = (X, T_\omega, V, \Phi)$ be the corresponding refined (infinite-valued) 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 semantics captured using the refined game is equivalent to the minimum infinite-valued model of the logic program.

Game Semantics of Intensional Logic Programming

We introduce a two-player game $\Gamma_P(C, w)$ during which, Player I has the role of the *Doubter* and Player II the role of the *Believer*.

The infinite tree T_ω of the game $\Gamma_P(C, w)$ consists of all the infinite sequences $\langle x_0, x_1, \ldots, x_k, \ldots\rangle$ which satisfy the following restrictions (in which A denotes a propositional atom, B_i denotes an intensional atom, u, v, z, y denote elements of W and S, S' denote subsets of W) for each $k \geq 0$:

$\mathbf{R_1}$: $x_0 = \langle C, w\rangle^-$.
$\mathbf{R_2}$: If $x_k = \langle \nabla A, u\rangle^-$, then $x_{k+1} = \langle A, S\rangle^+$, where $u \in ||\nabla||(S)$.
$\mathbf{R'_3}$: If $x_k = \langle A, S\rangle^+$ and $x_{k-1} = \langle \nabla A, u\rangle^-$, then either **(i)** $x_{k+1} = \langle A, v\rangle^-$, where $v \in S$, or **(ii)** $x_{k+1} = \langle A, S, S'\rangle^+$, where $S' \supset S$ and $u \notin ||\nabla||(S')$. A move of type **(ii)** will be called a *role switch*.
$\mathbf{R''_3}$: If $x_k = \langle A, S, S'\rangle^+$, then $x_{k+1} = \langle A, y\rangle^-$, where $y \in (S' - S)$.
$\mathbf{R_4}$: If $x_k = \langle A, v\rangle^-$, then $x_{k+1} = \langle \mathcal{C}, z\rangle^+$, where $\mathcal{C}$ is a clause in P of the form $B_0 \leftarrow B_1, \ldots, B_n$, such that either **(i)** $B_0 = A$ and $z = v$, or **(ii)** $B_0 = \nabla A$ and for every S satisfying $z \in ||\nabla||(S)$ it holds $v \in S$.
$\mathbf{R_5}$: If $x_k = \langle B_0 \leftarrow B_1, \ldots, B_n, z\rangle^+$, then $x_{k+1} = \langle B_j, z\rangle^-$, for some j with $1 \leq j \leq n$.
$\mathbf{R_6}$: If after x_k has been played, none of the above rules is applicable, then $x_{k+1} = \langle$*I've lost*$\rangle$.
$\mathbf{R_7}$: If $x_k = \langle$*I've lost*$\rangle$, then $x_{k+1} = \langle$*I've won*$\rangle$ (and vice-versa).

Some explanations are in order. Suppose that $C = \nabla A$ (the explanation for $C = A$ is similar). Initially, Player I plays the move $\langle \nabla A, w\rangle^-$. The intuitive explanation for this move is "*I doubt that ∇A is true in world w*". Player II

believes the truth of $\triangledown A$ in world w and for this reason he replies to the move of Player I with a pair $\langle A, S\rangle^+$, where $w \in ||\triangledown||(S)$. The explanation for this move is "*I believe that* $\triangledown A$ *is true in* w*; actually, I believe* A *is true in all the worlds contained in* S *and this implies that* $\triangledown A$ *is true in* w". Player I now responds with a pair $\langle A, v\rangle^-$ where $v \in S$. The intuition now is: "*I doubt that* A *is true in the world* v *of* S *(and therefore I continue to believe that* $\triangledown A$ *is not true in* w*)*". Player II must now establish that A is true in v. One way to achieve this is to use a clause with head A. A second (less direct) way is to prove that $\triangledown A$ holds at some world z with the property mentioned in Case (ii) of rule $\mathbf{R_4}$; this property guarantees that if $\triangledown A$ holds at z, then A holds at v. Therefore, Player II provides a pair $\langle \mathcal{C}, z\rangle^+$ where $\mathcal{C}$ is a program clause with head A or $\triangledown A$. If the head is A then z coincides with v; otherwise z is selected so that for all $S \subseteq W$ satisfying $z \in ||\triangledown||(S)$ it holds $v \in S$. The intuition is "*Using this rule and the context* z *I can establish that* A *is true in the world* v". Now Player I responds with a pair of the form $\langle B_i, z\rangle^-$, where B_i is one of the intensional atoms in the body of the rule that Player II has just played. The intuition is "*I doubt that* B_i *is true in world* z".

As the game proceeds, the two players may swap roles (the Believer may become Doubter and vice-versa). Suppose now that at some point of the game, the Believer replies to a move of the form $\langle \triangledown A, u\rangle^-$ of the Doubter by playing $\langle A, S\rangle^+$, where $u \in ||\triangledown||(S)$. The Doubter can respond in two different ways. His first option is to play $\langle A, v\rangle^-$ where $v \in S$. The intuition is: "*I doubt that* A *is true in the world* v *of* S *(and therefore I continue to believe that* $\triangledown A$ *is not true in* u*)*". Alternatively, the Doubter can play $\langle A, S, S'\rangle^+$ where $S' \supset S$ and $u \notin ||\triangledown||(S')$. The intuition here is "*I believe that the set of worlds where* A *is true is* $S' \supset S$ *and not* S *(as the Believer just claimed); since* $u \notin ||\triangledown||(S')$*, I was right in my belief that* $\triangledown A$ *is not true in* u". This second type of move has made the player that was a Doubter to become a Believer and his opponent to become a Doubter (*role-switch*). In move $\mathbf{R''_3}$, the new Doubter plays $\langle A, y\rangle^-$ where $y \in (S' - S)$. The intuition is "*I doubt that the set of worlds where* A *is true coincides with* S'*; more specifically, I doubt that* A *is true in the world* y".

The set of rewards is $D = \{0, \frac{1}{2}, 1\}$ i.e., a play of the game can be assigned the value 0 (Player I has won the play), value 1 (Player II has won), or the value $\frac{1}{2}$ (the result is a tie). Finally, the payoff function is defined as follows:

$$\Phi(s) = \begin{cases} 1, \text{ if Player II plays the } \langle \textit{I've won}\rangle \text{ move in } s \text{ or } s \text{ is a genuinely} \\ \quad \text{infinite play that contains an odd number of role-switches} \\ 0, \text{ if Player I plays the } \langle \textit{I've won}\rangle \text{ move in } s \text{ or } s \text{ is a genuinely} \\ \quad \text{infinite play that contains an even number of role-switches} \\ \frac{1}{2}, \text{ if } s \text{ contains an infinite number of role-switches} \end{cases}$$

According to the above definition, a player wins a play of the game if he manages either to play the $\langle \textit{I've won}\rangle$ move or to remain the doubter after a certain point of the play; otherwise the result of the play is a tie.

Theorem 4. *Let* P *be a program,* C *be a propositional or intensional atom and* $w \in W$*. Then, the game* $\Gamma_P(C, w)$ *is determined.*

Definition 4. *Let P be an intensional logic program. We define the* game interpretation *N_P of P such that for every propositional atom A that appears in P and for every $w \in W$, $N_P(A)(w)$ is equal to the value of the game $\Gamma_P(A, w)$.*

Definition 5. *Let P be an intensional logic program and assume that the two-valued denotations of all intensional operators in the heads of the clauses of P are universal, monotonic and conjunctive while the two-valued denotations of the intensional operators in the bodies of clauses are arbitrary functions in $\{0,1\}^W \to \{0,1\}^W$. We define the* game interpretation *N_P of P to be the interpretation which for every propositional atom A that appears in P and for every $w \in W$ has the property that $N_P(A)(w) = v$ if and only if the value of the game $\Gamma_P(A, w)$ is equal to v.*

Lemma 1. *Let P be a program and assume that the denotations of all intensional operators that appear in the heads of the clauses in P are universal, monotonic and conjunctive, while the denotations of intensional operators in the bodies of clauses are arbitrary functions in $\{0,1\}^W \to \{0,1\}^W$. Then, the game interpretation N_P of P is a model of P.*

Theorem 5. *Let P be a program and assume that the denotations of all intensional operators in the heads of the clauses of P are universal, monotonic and conjunctive while the denotations of the intensional operators in the bodies of clauses are arbitrary functions in $\{0,1\}^W \to \{0,1\}^W$. Then, the game interpretation N_P of P is a* minimal *model of P with respect to $\preceq$.*

If we restrict our focus to monotonic programs, we have a simpler version of the game, with two values (0, 1) and without role switches. This game is also proved to be determined and it gives a game interpretation that is a model of the program and is identical to the unique minimum model M_P of [24]:

Theorem 6. *Let P be an intensional logic program and assume that the denotations of all intensional operators in the heads of the clauses are universal, monotonic and conjunctive, and the denotations of all intensional operators that appear in the bodies of the clauses are monotonic. Then, the minimum intensional model M_P and the game interpretation N_P of P coincide.*

3 Conclusions

In this work we presented infinite-game semantics for logic programs. Initially, we proposed an infinite-game characterization of the well-founded semantics for function-free logic programs with negation. The game is a simple generalization of the standard game for negation-less logic programs introduced in [34] in which two players, the *Believer* and the *Doubter*, compete by trying to prove (respectively disprove) a query. The game for programs with negation that we proposed follows the same rules as the standard one, except that the players swap roles every time the play "passes through" negation. We showed 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. More specifically it is equivalent to the well-founded semantics of logic programming.

In order to prove that equivalence, we defined a refined version of the game, i.e. an infinite-valued game that uses infinite degrees of winning and losing for the two players. We then demonstrated that this refined game corresponds exactly to the infinite-valued minimum model semantics of negation of [28]. This implied that the unrefined game is equivalent to the well-founded semantics.

The study continued with *Intensional logic programming*, an extension of logic programming, which includes as special cases both *temporal* and *modal* logic programming. A new game was defined and shown to be determined and equivalent to the semantics of M. Orgun and W. W. Wadge [24] for the case of programs in which the denotations of intensional operators in the heads of the clauses are monotonic, universal and conjunctive and the denotations of intensional operators in the bodies of the clauses are monotonic.

We then extended the game to a three-valued one that also applies to programs with non-monotonic operators and showed its determinacy. We proved that this extended game provides minimal model semantics for intensional logic programs. This way we have introduced the first (to our knowledge) general semantic framework for non-monotonic intensional logic programming. The proposed game can be used as a yardstick in order to develop alternative semantical approaches for non-monotonic temporal and modal languages.

There are many aspects of this work that we feel that should be further investigated. First of all, the (unrefined) negation game could apply 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: the set V of truth values contains a F_α and a T_α for each countable ordinal α (see [28] 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 [31]). A first attempt towards this direction appears in [11].

A game semantics for (negation-free) disjunctive logic programming, similar to our approach, has recently been developed in [32]. It would be interesting to further broaden our understanding regarding the interplay between logic programming and game-theory, by extending the game semantics to apply to other logic programming languages since many recent results ([6, 4, 10, 9, 11, 32]) and the present work suggest that this is a fruitful avenue of research. For example, it would be desirable to devise a game semantics for answer-set programming.

The use of any new semantic approach for a programming language, can only be tested by its applications. It would therefore be interesting to apply

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.

References

1. Apt, K., Bol, R.: Logic Programming and Negation: A Survey. Journal of Logic Programming, 19,20:9–71 (1994)
2. Abramsky, S., McCusker, G.: 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. Baral, C., Gelfond, M.: Logic Programming and Knowledge Representation. Journal of Logic Programming, 19(20):73–148 (1994)
4. De Vos, M.: Logic Programming, Decisions and Games. PhD thesis, Vrije Universiteit Brussel (2001)
5. De Vos, M., Vermeir, D.: Choice logic programs and Nash equilibria in strategic games. In Computer Science Logic, pp. 266–276, Springer (1999)
6. Di Cosmo, R., Loddo, J. V., Nicolet, S.: A Game Semantics Foundation for Logic Programming. Proceedings of PLILP, LNCS 1490, 355–373 (1998)
7. Fariñas del Cerro, L.: MOLOG: A System that Extends PROLOG with Modal Logic. New Generation Computing, 4:35–50(1986)
8. Fitting, M.: Fixpoint Semantics for Logic Programming: A Survey. Theoretical Computer Science, 278(1-2):25–51 (2002)
9. Galanaki, Ch., Nomikos, Ch., Rondogiannis, P.: Game Semantics for Nonmonotonic Intensional Logic Programming. In: Logic Programming and Nonmonotonic Reasoning, Cabalar, P., Son, T. (Eds.) vol. 8148 of Lecture Notes in Computer Science, pp. 329-341, Springer Berlin Heidelberg (2013)
10. Galanaki, Ch., Rondogiannis, P., Wadge, W., W.: An Infinite-Game Semantics for Well-Founded Negation in Logic Programming. Annals of Pure and Applied Logic, 151(2–3):70–88 (2008)
11. Galanaki, Ch., Rondogiannis, P., Wadge, W., W.: General Logic Programs as Infinite Games. In: Topological and Game-Theoretic Aspects of Infinite Computations, Hertling, P., Selivanov, V., Thomas, W., Wadge, W., W., Wagner, K. (Eds.) no. 08271 in Dagstuhl Seminar Proceedings, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2008)
12. Gale, D., Stewart, F., M.: Infinite Games with Perfect Information. Annals of Mathematical Studies, 28:245–266 (1953)
13. Gallin D.: Intensional and Higher-Order Modal Logic: With Applications to Montague Semantics. American Elsevier Pub. Co. (1975)
14. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In: Proceedings of the Fifth Logic Programming Symposium, pages 1070–1080. MIT Press (1988)
15. Hintikka J., Sandu G.: Game-Theoretical Semantics. In: Handbook of Logic and Language (Second Edition), J. v. Benthem, A. t. Meulen, Eds. Elsevier (2011)
16. Martin, D., A.: Borel Determinacy. Annals of Math., 102:363–371 (1975)

17. Montague, R.: English as a Formal Language. In R. H. Thomason (Ed.), Formal Philosophy: Selected Papers of Richard Montague. Yale University Press, 108–221 (1974)
18. Moschovakis, Y., N.: Descriptive Set Theory. North-Holland (1980)
19. Moszkowski, B., C.: Executing Temporal Logic Programs. Seminar on Concurrency, 111–130 (1984)
20. Muskens, R.: Higher Order Modal Logic. In Handbook of Modal Logic, (P. Blackburn and J.F.A.K. van Benthem and F. Wolter eds.) Studies in Logic and Practical Reasoning, pp. 621-653, Dordrecht: Elsevier (2006)
21. Mycielski, J.: Games with Perfect Information. In R. J. Aumann, S. Hart (Eds.), Handbook of Game Theory, Elsevier, 41–70 (1992)
22. Nguyen, L., A.: MProlog: An Extension of Prolog for Modal Logic Programming. In: Proceedings of the 20th International Conference on Logic Programming, 469–470 (2004)
23. Orgun, M., A., P., Wadge, W., W.: Chronolog: A Temporal Logic Programming Language and its Formal Semantics, Technical Report, Department of Computer Science, University of Victoria, Canada (1988)
24. Orgun, M., A., P., Wadge, W., W.: Towards a Unified Theory of Intensional Logic Programming. Journal of Logic Programming, 13(4):413–440 (1992)
25. Przymusinska, H., Przymusinski, T.: Semantic Issues in Deductive Databases and Logic Programs. In: Banerji, R. (ed), Formal Techniques in Artificial Intelligence: a Source-Book, pages 321–367. North Holland (1990)
26. Przymusinski, T.C.: 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)
27. Rondogiannis, P., Gergatsoulis, M., Panayiotopoulos, T.: Branching-Time Logic Programming: The Language Cactus and its Applications. Computer Languages, 24(3):155-178 (1998)
28. Rondogiannis, P., Wadge, W. W.: Minimum Model Semantics for Logic Programs with Negation-as-Failure. ACM Transactions on Computational Logic, 6(2):441–467 (2005)
29. Rondogiannis, P., Wadge, W.W.: 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)
30. Scott, D.: Advice on Modal Logic. In K. Lambert (Ed.), Philosophical Problems in Logic, D. Reidel Publishing Company, 143–173 (1970)
31. Wadge, W. W.: Reducibility and Determinateness on the Baire Space. PhD thesis, University of California, Berkeley, 1984.
32. Tsouanas, T.: A game semantics for disjunctive logic programming. Annals of Pure and Applied Logic, vol.164, no. 11, pp. 1144-1175 (2013)
33. Ullman, J.: Database and Knowledge-Base Systems. Computer Science Press (1989)
34. van Emden, M., H.: Quantitative Deduction and its Fixpoint Theory. Journal of Logic Programming, 3(1):37–53 (1986)
35. van Gelder, A., Ross, K.A., Schlipf, J.S.: The Well-Founded Semantics for General Logic Programs. Journal of the ACM, 38(3):620–650 (1991)