General Logic Programs as Infinite Games[⋆]

Chrysida Galanaki[1], Panos Rondogiannis[1], and William W. Wadge[2]

[1] Department of Informatics & Telecommunications
University of Athens
Panepistimiopolis, 157 84 Athens, Greece
{chrysida,prondo}@di.uoa.gr
[2] Department of Computer Science
University of Victoria
PO Box 3055, STN CSC, Victoria, BC, Canada V8W 3P6
wwadge@csr.uvic.ca

Abstract. In [vE86] M.H. van Emden introduced a simple game semantics for definite logic programs[3]. Recently [RW05,GRW05], the authors extended this game to apply to logic programs with negation. Moreover, under the assumption that the programs have a finite number of rules, it was demonstrated in [RW05,GRW05] that the game is equivalent to the well-founded semantics of negation. In this paper we present work-in-progress towards demonstrating that the game of [RW05,GRW05] is equivalent to the well-founded semantics even in the case of programs that have a countably infinite number of rules. We argue however that in this case the proof of correctness has to be more involved. More specifically, in order to demonstrate that the game is correct one has to define a refined game in which each of the two players in his first move makes a bet in the form of a countable ordinal. Each ordinal can be considered as a kind of clock that imposes a "time limit" to the moves of the corresponding player. We argue that this refined game can be used to give the proof of correctness for the countably infinite case.

1 Introduction

In [vE86] M.H. van Emden first introduced a simple game semantics for definite logic programs. Recently [RW05,GRW05], the authors extended this game to apply to logic programs with negation. Moreover, under the assumption that the programs are finite, it is demonstrated in [RW05,GRW05] that the game semantics is equivalent to the so-called well-founded semantics of negation [vGRS91,Prz89]. It should be noted that the class of finite (propositional)

⋆ This research is supported by ΕΠΕΑΕΚ ΙΙ under the task "ΠΥΘΑΓΟΡΑΣ-ΙΙ: ΕΝΙΣΧΥΣΗ ΕΡΕΥΝΗΤΙΚΩΝ ΟΜΑΔΩΝ ΣΤΑ ΠΑΝΕΠΙΣΤΗΜΙΑ", Project title: *Applications of Computational Logic to the Semantic Web*, funded by the European Social Fund (75%) and the Greek Ministry of Education (25%).

[3] It is common in the theory of logic programming to study programs that are propositional and have a countable (possibly infinite) number of rules. We adopt this convention throughout this paper.

Dagstuhl Seminar Proceedings 08271
Topological and Game-Theoretic Aspects of Infinite Computations
http://drops.dagstuhl.de/opus/volltexte/20081651

programs is broad and important since, for example, every Datalog program with negation can be instantiated into such a program.

Of course, our ultimate goal would be to demonstrate that the game is also applicable to propositional programs that have a countably infinite number of rules. This would establish the applicability of the game in the most general way, since every first-order logic program with negation can be instantiated into a propositional logic program that has a countable (possibly infinite) number of rules (see for example [F02]). In this paper we present work-in-progress towards this goal. More specifically, we outline a proof technique which, as we argue, can be used to show the applicability of the game in this more general setting. The technique consists of defining a refined game, one in which the two players use bets in the form of countable ordinals as parts of their moves. We argue that this refined game is equivalent to the infinite-valued model [RW05] and therefore to the well-founded model. The basic ideas behind the game are outlined in the next section and formalized in the subsequent sections.

2 Logic Programming and Games: an Introduction

In this section we give an introduction to the game semantics of logic programming as described in [vE86,RW05,GRW05]. Moreover, we present at an intuitive level the main contribution of this paper.

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

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

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 $\mathit{literals}(P_G)$ for the set of all literals that appear in the goal clause G or in the bodies of clauses of P; similarly, we write $\mathit{negvars}(P)$ for the set of all propositional variables, that appear in negative literals in the bodies of clauses of P. A logic program is called *definite* if none of its rules contains negative literals.

Let P be a definite logic program and G a goal clause. In the van Emden game there exist two players, Player I and Player II who try to prove (respectively disprove) the goal G. Player I, the *Believer*, believes that G will succeed and his first move is to play G. Player II, the *Doubter*, thinks G will fail. His first move is to choose the (only) variable that exists in G (declaring in this way his belief that this variable will fail). From then on play proceeds as follows: the Believer 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 fact. For the Doubter, this means choosing a variable for which there is no rule in the program. Finally, the Doubter has an important advantage: he wins if the game never ends.

The above game can be extended to apply to logic programs that allow negative literals in clause bodies. Actually, the game introduced in [RW05,GRW05] simply adds one extra rule to the van Emden game: when one of the players plays a literal of the form $\sim p$ then his opponent must, on the next move, play the atom p. The intuition behind this new rule is that when negation is encountered, the two players swap roles: the believer becomes the doubter and vice-versa; the situation will be called a *role-switch* from now on. As before, any player who has no legal move loses. If on the other hand the game play is infinite and after a certain point one of the players remains a doubter, he wins. Finally, if the two players swap roles infinitely often, the result is a tie.

Notice that in the above discussion the fact that the program P may be infinite, plays no role. However, when one tries to establish that the game is equivalent to the well-founded semantics of negation, it is convenient to make the assumption that the program P is finite (see [RW05,GRW05]). In the rest of this section we will explain the reasons why until now we have restricted attention to finite programs and we will intuitively explain how one can lift the proofs to the more general case. The rest of the paper will formalize the intuitive ideas that we will now present.

The well-founded semantics of logic programs is based on a three-valued logic, namely a logic that uses the truth values *False*, 0 and *True*. Intuitively, in [RW05,GRW05] it was demonstrated 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 that has this atom in the goal clause; moreover, the value 0 was shown to correspond to the case where the best choice for both players is to lead the game to a tie. It is well-known that 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.

The refinement of the well-founded model that we use is the characterization that two of the authors have recently obtained [RW02,RW05]. More specifically, the *infinite-valued semantics* introduced in [RW02,RW05] is a refinement of the well-founded semantics and it uses instead an infinite number of truth values ordered as follows:

$$F_0 < F_1 < F_2 < \cdots < F_\alpha < \cdots < 0 < \cdots < T_\alpha < \cdots < T_2 < T_1 < T_0$$

Notice that the above set of truth values has an F_α and a T_α for every countable ordinal α. However, it can be shown that when we are dealing with finite logic programs we only need F_k and T_k for $k < \omega$.

Suppose now we are dealing with finite programs. Motivated by the infinite-valued semantics, we can define a refined game which supports different degrees of winning and losing. More specifically, assume we are given a play that is a win for Player I. In order to calculate the payoff that corresponds to this play, we simply count the number of role-switches that have taken place; if this number is equal to say k then the payoff of the play is taken to be equal to T_k. Then, it can be shown (see [RW05,GRW05] for the details) that the refined game is equivalent to the infinite-valued semantics, which immediately implies that the unrefined game is equivalent to the well-founded semantics.

But how can we extend this proof idea to the case of infinite programs? Notice that the infinite-valued semantics for infinite programs may assign to atoms values of the form T_α (or F_α), where α is an infinite ordinal. How can we now define the refined game? Counting the number of role-switches in a play of the game is no longer enough because this process can only return a natural number and not a countable ordinal. In other words, we want to somehow express the fact that if one of the players can win the game, then the time that will be required can be measured by a countable ordinal.

In this paper we argue that the above problem can be overcome by adopting a technique that is common in the theory of infinite games (see for example [Wad84][page 47]). We define a refined game in which the two players make a bet in the form of a countable ordinal as part of their first move. The new game then has the following extra rule:

> During a role-switch both players have to decrease the value of their ordinal, if this is greater than 0. In every other move, the ordinal must be identical to the ordinal played by the same player in his previous move.

Now, Player I wins an extended game iff he manages to win the game (in the original sense) and his initial bet has not reduced to 0 before the last role-switch has taken place. Similarly for Player II. Each bet can be considered as a type of clock which imposes a "time limit" on the corresponding player.

The correctness of the original game for infinite propositional programs can then be established by showing that this refined game is equivalent to the infinite-valued semantics.

The rest of the paper is organized as follows: Section 3 describes the game for logic programs with negation (this section is included for reasons of completeness, since the game is actually the one introduced in [RW05,GRW05]). Section 4 defines the refined game in which the players use bets. Section 5 provides a sketch of the proof that the game is equivalent to the well-founded semantics of logic programs. Finally, Section 6 concludes the paper.

3 Game Semantics for Infinite Programs

The semantics that we develop in this paper are based on the so-called *infinite games of perfect information* (or *PI-games* for short) [GS53,Myc92]. The games will take place between two players that we will call *Player I* and *Player II*:

Definition 1. *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 set of* rules *which impose 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 2. *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:*

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

A (complete) play *of the game 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 notion that plays a very important role in the theory of infinite games is that of determinacy:

Definition 3. *A game $\Gamma = (X, R, A, B, D, \Phi)$ is* determined *with* value v *if*

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

We now 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 corresponding PI-game $\Gamma_{P_G} = (X, R, A, B, D, \Phi)$, as follows:

3.1 The set of moves

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

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

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) a propositional variable that appears in a negative literal in the body of some clause of P.

3.2 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. These two moves constitute a role-switch.

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.

3.3 The sets of strategies

The sets of strategies for the game are specified as in Definition 1.

3.4 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.

3.5 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. The following two definitions will be useful in defining the payoff function:

Definition 4. *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 5. *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.*

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.

4 The Refined Negation Game

In this section we give a precise definition of the refined negation game. Let P be a logic program and G a goal clause. We define a corresponding PI-game $\Gamma_{P_G} = (X, R, A, B, D, \Phi)$, as follows:

4.1 The set of moves

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

$$X = \{(x, \alpha) : x \in \{G\} \cup P \cup \mathit{literals}(P_G) \cup \mathit{negvars}(P), \alpha \text{ is a countable ordinal}\}$$

In other words, the moves are now pairs: the first part of each pair is as in the unrefined game; the second part is a countable ordinal.

4.2 The rules of the game

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

- (R1) The first move of Player I consists of the goal clause $\leftarrow p$, together with a countable ordinal, which we call the *bet* of Player I. The first move of Player II consists of the atom p together with a (possibly different) countable ordinal, which we call the *bet* of Player II.
- (R2) If the first part of the previous move is a clause, the first part of the next move is one of the literals in the body of the clause.
- (R3) If the first part of the previous move is a positive literal p, the first part of the next move is a clause in P whose head is p.
- (R4) If the first part of the previous move is a negative literal $\sim p$, the first part of the next move must be p itself. These two moves constitute a *role-switch*.
- (R5) During a role switch both players have to decrease the value of their ordinal, if this is greater than 0. In every other move, the ordinal must be identical to the ordinal played by the same player in his previous move.

Rule violations are defined analogously as in the unrefined case.

4.3 The sets of strategies

The sets of strategies for the game are specified as in the case of the unrefined game.

4.4 The set of rewards

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

4.5 The payoff function

The notions of true-play and false-play are identical to the ones used in the unrefined case. Consider now a play s of the refined game. 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 there are no rule violations during the game. Moreover, we define $\| \ s \ \|_I$ (respectively $\| \ s \ \|_{II}$) to be equal to the initial bet of Player I (respectively Player II) in $\hat{s}$. Finally, we will say that the *flag has fallen* for one of the players in $\hat{s}$ if this player's ordinal has decreased to 0 before the last role-switch in $\hat{s}$.

The refined payoff function Φ is then defined as follows:

$$\Phi(s) = \begin{cases} T_{\|s\|_I}, & \text{if } s \text{ is a true-play and Player's I flag has not fallen in } \hat{s} \\ F_{\|s\|_{II}}, & \text{if } s \text{ is a false-play and Player's II flag has not fallen in } \hat{s} \\ 0, & \text{otherwise} \end{cases}$$

4.6 An Example

We can now illustrate the above definitions with the following example:

Example 1. Consider the countably infinite program P:

$$\begin{array}{l|l|l} q_0 \leftarrow & q \leftarrow q_1 & p \leftarrow\!\sim q \\ q_1 \leftarrow\!\sim q_0 & q \leftarrow q_3 & \\ q_2 \leftarrow\!\sim q_1 & q \leftarrow q_5 & \\ q_3 \leftarrow\!\sim q_2 & & \\ \vdots & \quad\vdots & \end{array}$$

and the goal $G = \leftarrow p$. It can be easily seen that the infinite-valued semantics assigns to the variable p the value $T_{\omega+1}$. We claim that this is also the value of the game that has G as its first move. Consider a play of the following form (notice that in this example the ordinals played by Player II are irrelevant to the calculation of the payoff of the play as long as they obey rule R5 of the game):

Player I		*Player II*	
$\leftarrow p$	$\omega+1$	p	$\omega+1$
$p \leftarrow\sim q$	$\omega+1$	$\sim q$	3
q	ω	$q \leftarrow q_3$	3
q_3	ω	$q_3 \leftarrow\sim q_2$	3
$\sim q_2$	2	q_2	2
$q_2 \leftarrow\sim q_1$	2	$\sim q_1$	1
q_1	1	$q_1 \leftarrow\sim q_0$	1
$\sim q_0$	0	q_0	0
$q_0 \leftarrow$	0	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Player II is forced to break the rules first, so this is a true-play. The payoff for the play is $T_{\omega+1}$ since the initial bet of Player I is $\omega+1$. In fact this is the best payoff that Player I can get. If he plays ω as his initial bet, after the first role-switch he will have to reduce his initial bet to a natural number n. But then Player II can choose a rule $q \leftarrow q_m$ such that $m > n$. Eventually Player's I flag will fall and the payoff will be 0.

Player I		*Player II*	
$\leftarrow p$	ω	p	$\omega+1$
$p \leftarrow\sim q$	ω	$\sim q$	m
q	n	$q \leftarrow q_m$	m
q_m	n	$q_m \leftarrow\sim q_{m-1}$	m
$\sim q_{m-1}$	$n-1$	q_{m-1}	$m-1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$q_2 \leftarrow\sim q_1$	1	$\sim q_1$	3
q_1	0	$q_1 \leftarrow\sim q_0$	3
$\sim q_0$	0	q_0	2
$q_0 \leftarrow$	0	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

The above discussion leads to the conclusion that in this example the game agrees with the infinite-valued model (and consequently with the well-founded model). A formalization of this observation is given in the next section.

5 Equivalence with the Well-Founded Semantics

In this section we outline how the proof of the equivalence of the refined game to the infinite-valued semantics can be established. The material in this section represents work-in-progress. We intend to provide a more detailed account of the statements in this section in the final version of the paper.

The first one of the statements concerns the determinacy of the refined negation game:

Theorem 1. *Let P be a program, G a goal clause and let Γ_{P_G} be the corresponding refined game. Then, Γ_{P_G} is determined.*

The proof of the above can be given based on the same ideas as the corresponding proof in [GRW05]. Intuitively, one uses results from Borel determinacy of win-lose games in order to establish the determinacy of this more complicated type of games.

The second statement concerns the equivalence of the refined game to the infinite-valued 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 $\Gamma_{P_G} = (X, R, A, B, V, \Phi)$ be the corresponding refined game. Moreover, let M_P be the minimum infinite-valued model of P. Then, Γ_{P_G} has value T_α (respectively F_α) if and only if $M_P(p) = T_\alpha$ (respectively $M_P(p) = F_\alpha$).*

Again, the proof of the above can be performed along the same lines as in [GRW05]. However, the proof must now use transfinite induction (instead of ordinary induction as in [GRW05]). The main idea is therefore that one shows inductively that for every level of truth values, the refined game characterization and the infinite-valued one coincide.

6 Conclusions

In [RW05,GRW05] we introduced a game for logic programs with negation and demonstrated that if one restricts attention to finite such programs, the game is equivalent to the well-founded semantics. In this paper we have outlined how one can prove the above equivalence even for infinite logic programs.

Future work includes presenting the proofs of the previous section in full detail and using the game to prove the correctness of program transformations. It is our hope that the game-theoretic work that is just starting to emerge in the area of logic programming, will eventually prove to be equally valuable as the corresponding work in functional programming.

References

[F02] M. Fitting. Fixpoint Semantics for Logic Programming: A Survey. *Theoretical Computer Science* 278(1–2), 25–51, 2002.

[GRW05] Ch. Galanaki, P. Rondogiannis and W.W. Wadge. An Infinite-Game Semantics for Well-Founded Negation in Logic Programming. Submitted to *Annals of Pure and Applied Logic* (2005).

[GS53] 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.

[Myc92] 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.

[Prz89] 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.

[RW02] 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. (available from `http://www.di.uoa.gr/∼prondo/inf.ps`).

[RW05] 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.

[RW05] P. Rondogiannis and W.W. Wadge. An Infinite-Game Semantics for Negation in Logic Programming. In *Proceedings of the first International Workshop on Games for Logic and Programming Languages (GaLoP)*, pages 77–91. Edinburgh, April 2005.

[vE86] M.H. van Emden. Quantitative Deduction and its Fixpoint Theory. *Journal of Logic Programming*, 3(1):37–53, 1986.

[vGRS91] 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.

[Wad84] W.W. Wadge. *Reducibility and Determinateness on the Baire Space*. PhD thesis, University of California, Berkeley, 1984.