Theory of Logic Programming and the Semantics of Non-Monotonic Formal Grammars

Vassileios Kountouriotis[*]

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
b.kountouriotis@gmail.com

Abstract

In this dissertation we give the first proper mathematical treatment of the semantics of Boolean grammars. We show that for every Boolean grammar there exists a distinguished (three-valued) interpretation of the non-terminal symbols, which satisfies all the rules of the grammar and at the same time is the least fixed-point of an operator associated with the grammar. Then, we propose an $\mathcal{O}(n^3)$ algorithm for parsing that applies to any normalized Boolean grammar. Finally, we give a purely game-theoretic characterization of Boolean grammars: we propose a two-player infinite game of perfect information for Boolean grammars, which is equivalent to their well-founded semantics.

1 Well-Founded Semantics for Boolean Grammars

In this dissertation we propose and investigate the properties of the well-founded semantics for Boolean grammars. We begin by giving the formal definition of Boolean grammars. We then define the well-founded semantics of a given Boolean grammar as the least fixed-point of an operator associated with the grammar. Moreover we examine the mathematical properties of this operator in detail.

1.1 Syntax

In [Okh04] A. Okhotin proposed the class of Boolean grammars. Formally:

[*]Dissertation Advisor: Panagiotis Rondogiannis, Associate Professor

Definition 1.1. *A Boolean grammar is a quadruple $G = (\Sigma, N, P, S)$, where Σ and N are disjoint finite non-empty sets of terminal and non-terminal symbols respectively, P is a finite set of rules, each of the form*

$$A \to \alpha_1 \& \cdots \& \alpha_m \& \neg\beta_1 \& \cdots \& \neg\beta_n \qquad (m + n \geq 1, \alpha_i, \beta_j \in (\Sigma \cup N)^*),$$

and $S \in N$ is the start symbol of the grammar. We will call the non-terminal A the head *of the rule, the α_i's* positive conjuncts *and the $\neg\beta_j$'s* negative *ones.*

We will often use the short notation $A \to \varphi_1 \mid \cdots \mid \varphi_k$ to represent k rules of the form $A \to \varphi_i$.

1.2 Interpretations and Models for Boolean Grammars

Definition 1.2. *Let Σ be a finite non-empty set of symbols. Then, a (three-valued) language over Σ is a function from Σ^* to the set $\left\{0, \frac{1}{2}, 1\right\}$.*

Intuitively, given a three-valued language L and a string w over the alphabet of L, there are three-cases: either $w \in L$ (i.e., $L(w) = 1$), or $w \notin L$ (i.e., $L(w) = 0$), or finally, the membership of w in L is unclear (i.e., $L(w) = \frac{1}{2}$). Given this extended notion of language, it is now possible to interpret the grammar $S \to \neg S$: its meaning is the language which assigns to every string the value $\frac{1}{2}$.

The following definition, which generalizes the familiar notion of concatenation of languages, will be used in the rest of the dissertation:

Definition 1.3. *Let Σ be a finite non-empty set of symbols and let $L_1, \ldots, L_n$ be (three-valued) languages over Σ. We define the* three-valued concatenation *of the languages $L_1, \ldots, L_n$ to be the language L such that for every $w \in \Sigma^*$:*

$$L(w) = \max_{\substack{(w_1, \ldots, w_n): \\ w = w_1 \cdots w_n}} \left(\min_{1 \leq i \leq n} L_i(w_i) \right)$$

The concatenation of $L_1, \ldots, L_n$ will be denoted by $L_1 \circ \cdots \circ L_n$.

It can be easily checked that when the languages involved are total (i.e., with no $\frac{1}{2}$ values assigned to strings) then the above definition reduces to the familiar definition of concatenation.

We can now define the notion of *interpretation* of a given Boolean grammar:

Definition 1.4. *An* interpretation *I of a Boolean grammar $G = (\Sigma, N, P, S)$ is a function $I : N \to \left(\Sigma^* \to \left\{0, \frac{1}{2}, 1\right\}\right)$.*

An interpretation I can be recursively extended to apply to expressions that appear in the right-hand sides of Boolean grammar rules:

Definition 1.5. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and let I be an interpretation of G. Then, the extension $\widehat{I}$ of I is defined as follows:*

- *For every $w \in \Sigma^*$, it is $\widehat{I}(\epsilon)(w) = 1$ if $w = \epsilon$, and $\widehat{I}(\epsilon)(w) = 0$ otherwise.*
- *Let $A \in N$. Then, for every $w \in \Sigma^*$, it is $\widehat{I}(A)(w) = I(A)(w)$.*
- *Let $a \in \Sigma$. Then, for every $w \in \Sigma^*$, it is $\widehat{I}(a)(w) = 1$ if $w = a$, and $\widehat{I}(a)(w) = 0$ otherwise.*
- *Let $\alpha = \alpha_1 \cdots \alpha_n$, $n \geq 2$, $\alpha_i \in \Sigma \cup N$. Then, for every $w \in \Sigma^*$, it is $\widehat{I}(\alpha)(w) = (\widehat{I}(\alpha_1) \circ \cdots \circ \widehat{I}(\alpha_n))(w)$.*
- *Let $\alpha \in (\Sigma \cup N)^*$. Then, for every $w \in \Sigma^*$, it is $\widehat{I}(\neg\alpha)(w) = 1 - \widehat{I}(\alpha)(w)$.*
- *Let $l_1, \ldots, l_n$ be conjuncts. Then, for every $w \in \Sigma^*$, it is $\widehat{I}(l_1 \& \cdots \& l_n)(w) = \min\{\widehat{I}(l_1)(w), \ldots, \widehat{I}(l_n)(w)\}$.*

We are now in a position to define the notion of a model of a Boolean grammar:

Definition 1.6. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and I an interpretation of G. Then, I is a* model *of G if for every rule $A \to l_1 \& \cdots \& l_n$ in P and for every $w \in \Sigma^*$, it is $I(A)(w) \geq \widehat{I}(l_1 \& \cdots \& l_n)(w)$.*

In the definition of the well-founded model, two orderings on interpretations play a crucial role (see [PP90] for the corresponding ordering in the case of logic programming). Given two interpretations, the first ordering (usually called the *standard ordering*) compares their *degree of truth*:

Definition 1.7. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and I, J be two interpretations of G. Then, we write $I \preceq J$ if for all $A \in N$ and for all $w \in \Sigma^*$, $I(A)(w) \leq J(A)(w)$.*

The following lemma is easy to establish:

Lemma 1.8. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and I, J be two interpretations of G such that $I \preceq J$. Then, for all $\alpha \in (\Sigma \cup N)^*$ and for all $w \in \Sigma^*$, $\widehat{I}(\alpha)(w) \leq \widehat{J}(\alpha)(w)$.*

Among the interpretations of a given Boolean grammar, there is one which is the least with respect to the $\preceq$ ordering and is denoted by $\perp$. That is, for all A and all w, $\perp(A)(w) = 0$.

The second ordering (usually called the *Fitting ordering*) compares the *degree of information* of two interpretations. We first need to define the corresponding numerical ordering:

Definition 1.9. *Let $v_1, v_2 \in \{0, \frac{1}{2}, 1\}$. We write $v_1 \leq_F v_2$ if and only if either $v_1 = v_2$ or $v_1 = \frac{1}{2}$.*

Definition 1.10. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and I, J be two interpretations of G. Then, we write $I \preceq_F J$ if for all $A \in N$ and for all $w \in \Sigma^*$, $I(A)(w) \leq_F J(A)(w)$.*

We now establish a lemma regarding $\preceq_F$ which is similar to Lemma 1.8 for $\preceq$:

Lemma 1.11. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and I, J be two interpretations of G such that $I \preceq_F J$. Then, for any conjunct l (either positive or negative) and for any $w \in \Sigma^*$, $\widehat{I}(l)(w) \leq_F \widehat{J}(l)(w)$.*

Among the interpretations of a given Boolean grammar, there is one which is the least with respect to the $\preceq_F$ ordering and is denoted by $\perp_F$. That is, for all A and all w, $\perp_F (A)(w) = \frac{1}{2}$.

Given a set U of interpretations, we will write $lub_{\preceq}U$ for the least upper bound of the members of U under the standard ordering. Formally:

$$(lub_{\preceq}U)(A)(w) = \begin{cases} 1, \text{if there exists } I \in U \text{ such that } I(A)(w) = 1 \\ 0, \text{if for all } I \in U,\ I(A)(w) = 0 \\ \frac{1}{2}, \text{otherwise} \end{cases}$$

The situation changes when one wants to define $lub_{\preceq_F}U$, that is, the least upper bound of the members of U under the Fitting ordering, since this notion cannot in general be defined for an arbitrary set of interpretations U. However, $lub_{\preceq_F}U$ can be defined if U is a directed set of interpretations, i.e., if for every $I_1, I_2 \in U$ there exists $J \in U$ such that $I_1 \preceq_F J$ and $I_2 \preceq_F J$. In this case $lub_{\preceq_F}U$ is defined as follows:

$$(lub_{\preceq_F}U)(A)(w) = \begin{cases} 1, \text{if there exists } I \in U \text{ such that } I(A)(w) = 1 \\ 0, \text{if there exists } I \in U \text{ such that } I(A)(w) = 0 \\ \frac{1}{2}, \text{otherwise} \end{cases}$$

1.3 Well-Founded Semantics for Boolean Grammars

In this section we define the well-founded semantics of Boolean grammars. The basic idea behind the well-founded semantics is that the intended model of the grammar is constructed in stages that are related to the levels of negation used by the grammar. At each step of this process and for every non-terminal symbol, the values of certain strings are computed and fixed (as either true or false); at each new level, the values of more and more strings become fixed (and this is a monotonic procedure in the sense that values of strings that have been fixed for a given non-terminal in a previous stage, are not altered by the next stages). At the end of all the stages, certain strings for certain non-terminals may have not managed to get the status of either true or false (this will be due to circularities through negation in the grammar). Such strings are classified as unknown (i.e., $\frac{1}{2}$).

1.4 The Properties of Θ_J

Consider the Boolean grammar $G = (\Sigma, N, P, S)$. Then, for any interpretation J of G we define the operator $[\Theta_G]_J : \mathcal{I} \to \mathcal{I}$ on the set $\mathcal{I}$ of all 3-valued interpretations of G.

Definition 1.12. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar, let $\mathcal{I}$ be the set of all three-valued interpretations of G and let $J \in \mathcal{I}$. The operator $[\Theta_G]_J : \mathcal{I} \to \mathcal{I}$ is defined as follows. For every $I \in \mathcal{I}$, for all $A \in N$ and for all $w \in \Sigma^*$:*

1. *$[\Theta_G]_J(I)(A)(w) = 1$ if there exists a rule $A \to l_1 \& \cdots \& l_r$ in P such that for every positive l_i it is $\widehat{I}(l_i)(w) = 1$ and for every negative l_i it is $\widehat{J}(l_i)(w) = 1$;*
2. *$[\Theta_G]_J(I)(A)(w) = 0$ if for every rule $A \to l_1 \& \cdots \& l_r$ in P, either there exists a positive l_i such that $\widehat{I}(l_i)(w) = 0$, or there exists a negative l_i such that $\widehat{J}(l_i)(w) = 0$;*
3. *$[\Theta_G]_J(I)(A)(w) = \frac{1}{2}$, otherwise.*

An important fact regarding the operator $[\Theta_G]_J$ is that it is monotonic with respect to the $\preceq$ ordering of interpretations:

Lemma 1.13. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and let J be an interpretation of G. Then, the operator $[\Theta_G]_J$ is monotonic with respect to the $\preceq$ ordering of interpretations.*

The next definition and theorem demonstrate that $[\Theta_G]_J$ has a unique least fixed-point:

Definition 1.14. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and let J be an interpretation of G. Define:*

$$\begin{array}{lcl} [\Theta_G]_J^{\uparrow 0} & = & \perp \\ [\Theta_G]_J^{\uparrow n+1} & = & [\Theta_G]_J([\Theta_G]_J^{\uparrow n}) \\ [\Theta_G]_J^{\uparrow \omega} & = & lub_{\preceq}\{[\Theta_G]_J^{\uparrow n} \mid n < \omega\} \end{array}$$

Theorem 1.15. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and let J be an interpretation of G. Then, the sequence $\{[\Theta_G]_J^{\uparrow n}\}_{n<\omega}$ is increasing with respect to $\preceq$ and $[\Theta_G]_J^{\uparrow \omega}$ is the unique least fixed-point of the operator $[\Theta_G]_J$ with respect to $\preceq$.*

1.5 The Properties of Ω_G

We will denote by $\Omega_G(J)$ the least fixed-point $[\Theta_G]_J^{\uparrow \omega}$ of $[\Theta_G]_J$. Given a grammar G, we can use the Ω_G operator to construct a sequence of interpretations whose least upper bound M_G (with respect to $\preceq_F$) will prove to be a distinguished model of G.

The definition of M_G has as follows:

Definition 1.16. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar. Define:*

$$\begin{array}{lcl} M_{G,0} & = & \perp_F \\ M_{G,n+1} & = & \Omega_G(M_{G,n}) \\ M_G & = & lub_{\preceq_F}\{M_{G,n} \mid n < \omega\} \end{array}$$

As we are going to see shortly, the operator Ω_G is monotonic with respect to $\preceq_F$ and this ensures that the sequence $\{M_{G,n}\}_{n<\omega}$ is increasing (which ensures that $lub_{\preceq_F}$ is well-defined).

Lemma 1.17. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar. Then, Ω_G is monotonic with respect to the $\preceq_F$ ordering of interpretations.*

Theorem 1.18. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar. Then, the sequence $\{M_{G,n}\}_{n<\omega}$ is increasing with respect to the $\preceq_F$ ordering of interpretations. Moreover, M_G is the least fixed-point of the operator Ω_G.*

1.6 The Well-Founded Model M_G

The above results lead to the following theorem, which demonstrates that M_G satisfies all the rules of the grammar G:

Theorem 1.19. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar. Then, M_G is a model of G (which will be called the* well-founded model of *G).*

We now give an example that illustrates the well-founded construction as this has been defined above:

1.7 An Undecidability Result

At this point we examine a natural question that springs to mind after the introduction of the three-valued well-founded model. Since most of the current work in formal language theory is based on two-valued languages, it is reasonable to wonder whether the problem "Given a Boolean grammar G, is M_G two-valued?" is decidable. The following theorem demonstrates that this is not the case.

Theorem 1.20. *The following problem is undecidable: "Given a Boolean grammar $G = (\Sigma, N, P, S)$, decide whether for all $w \in \Sigma^*$, $M_G(S)(w) \in \{0, 1\}$".*

1.8 Normal Form

The binary normal form we will derive is defined as follows:

Definition 1.21. *A Boolean grammar $G = (\Sigma, N \cup \{U, T\}, P, S)$ is said to be in binary normal form if P contains the rules $U \rightarrow \neg U$ and $T \rightarrow \neg\epsilon$,* *where U and T are two special symbols not in N, and every other rule in P is of the form:*

$$\begin{array}{lll} A & \rightarrow & B_1C_1\&\cdots\&B_mC_m\&\neg D_1E_1\&\cdots\&\neg D_nE_n\&TT[\&U] \quad (m,n \geq 0) \\ A & \rightarrow & a[\&U] \\ S & \rightarrow & \epsilon[\&U] \quad \textit{(only if S does not appear in right-hand sides of rules)} \end{array}$$

where $A, B_i, C_i, D_j, E_j \in N$, $a \in \Sigma$, and the brackets denote an optional part.

Theorem 1.22. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar. Then there exists a grammar $G' = (\Sigma, N', P', S)$ in binary normal form such that $M_G(S) = M_{G'}(S)$.*

The proof of Theorem 1.22 is based on the definition of certain meaning-preserving grammar transformations.

The normal form we derive, generalizes the well-known Chomsky normal form for context-free grammars as-well-as the binary normal form for Boolean grammars introduced in [Okh04]. Actually, certain of the steps we adopt, were initially proposed in [Okh04], the main difference being that the binary normal form obtained there, always produces two-valued Boolean languages.

The steps of the proposed procedure, can be summarized as follows:

- The initial Boolean grammar is first brought into *pre-normal form*. This is just a simpler and more manageable form of the initial grammar.
- The grammar is then transformed into *direct form*. This means that if a non-terminal of the previous form of the grammar could produce a string of length one (possibly through the use of many rules), then this fact is recorded by using a single rule in the new grammar. The same happens even if the status of the string of length one was undefined in the previous grammar.
- The next step is to bring the grammar into an *ϵ-free form*, i.e., a form in which no non-terminal produces the string ϵ.
- The final step is to bring the grammar into a *binary normal form*, i.e., a form in which the "long" rules of the grammar contain conjuncts which consist of two non-terminals (with the possible exception of the non-terminal U, see Definition 1.21 above).

1.9 Parsing under the Well-Founded Semantics

We next present an algorithm that computes the truth value of the membership of an input string $w \neq \epsilon$ in the language defined by a grammar G, which is assumed to be in binary normal form.

Algorithm for parsing under $G = (\Sigma, N, P, S)$

Input: string $w = a_1 \cdots a_n \in \Sigma^+$

Initialization step:

```
for $i := 1$ to $n$ do begin
    for every $A \in N$ do
        if there exists a rule $A \to a_i$ then $M[A, i, i] := 1$
        else if there exists a rule $A \to a_i \& U$ then $M[A, i, i] := \frac{1}{2}$
        else $M[A, i, i] := 0$
end
```

Main loop:

```
for $d := 2$ to $n$ do
    for $i := 1$ to $n - d + 1$ do begin
        $j := i + d - 1$
        for every $B, C \in N$ such that $BC$ appears in the right-hand side of a rule do
            $Q[B, C, i, j] := \max_{\ell=i}^{j-1} \min\{M[B, i, \ell], M[C, \ell + 1, j]\}$
        for every $A \in N$ do $M[A, i, j]$:=0
        for every rule $A \to B_1C_1\& \ldots \&B_mC_m\&\neg D_1E_1\& \ldots \&\neg D_rE_r\&TT\&U$ do begin
            $v := \min\{\frac{1}{2}, \min_{p=1}^{m} Q[B_p, C_p, i, j], \min_{q=1}^{r}(1 - Q[D_q, E_q, i, j])\}$
            if $v > M[A, i, j]$ then $M[A, i, j] := v$
        end
        for every rule $A \to B_1C_1\& \ldots \&B_mC_m\&\neg D_1E_1\& \ldots \&\neg D_rE_r\&TT$ do begin
            $v := \min\{\min_{p=1}^{m} Q[B_p, C_p, i, j], \min_{q=1}^{r}(1 - Q[D_q, E_q, i, j])\}$
            if $v > M[A, i, j]$ then $M[A, i, j] := v$
        end
    end
return $M[S, 1, n]$
```

The correctness of the above algorithm is established by the following theorem:

Theorem 1.23. *Let $G = (\Sigma, N, P, S)$ be a fixed Boolean grammar. Then, for every string $w = a_1 \cdots a_n \in \Sigma^+$, the above algorithm computes the correct value $M_G(A)(w)$, in time $\mathcal{O}(n^3)$.*

2 Game-Theoretic Semantics for Boolean Grammars

In this section we demonstrate that there exists a simple game-theoretic characterization of the semantics of Boolean grammars. In particular, we prove that this game-theoretic approach is equivalent to the well-founded semantics of the grammar.

2.1 A Formalization of the Game

In this section we formalize the game we have just described. In particular, we present some basic background on infinite games of perfect information, which we then use in order to present the proposed game for Boolean grammars in a formal way.

Definition 2.1. *An infinite game of perfect information is a quadruple* $\Gamma = (X, R, D, \Phi)$*, where:*

- X *is a nonempty 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.*
- 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*.*
- $\Phi : \{\langle x_0, x_1, \ldots \rangle \in X^\omega \mid \forall i \langle x_0, x_1, \ldots, x_i \rangle \in R\} \to D$*, is the* payoff function *of the game.*

Based on the set of moves X of a game, we define two sets $\text{Strat}^I(\Gamma)$ and $\text{Strat}^{II}(\Gamma)$ which correspond to the set of strategies for Player I and Player II respectively. A strategy $\sigma \in \text{Strat}^I(\Gamma)$ assigns a move to each even length partial play of the game; similarly for $\tau \in \text{Strat}^{II}(\Gamma)$ and odd length partial plays.

Definition 2.2. *Let* $\Gamma = (X, R, D, \Phi)$ *be a game. Define the set of strategies for Player I and Player II respectively to be the sets:*

- $Strat^I(\Gamma) = \{f \in (\bigcup_{n<\omega} X^{2n}) \to X \mid \langle x_0, \ldots, x_{n-1}, f(\langle x_0, \ldots, x_{n-1} \rangle) \rangle \in R\}$
- $Strat^{II}(\Gamma) = \{f \in (\bigcup_{n<\omega} X^{2n+1}) \to X \mid \langle x_0, \ldots, x_{n-1}, f(\langle x_0, \ldots, x_{n-1} \rangle) \rangle \in R\}$

Definition 2.3. *Let* Γ *be a game and let* $\sigma \in Strat^I(\Gamma)$ *and* $\tau \in Strat^{II}(\Gamma)$*. We define the following sequence:*

$$\begin{aligned} s_0 &= \sigma(\langle\rangle) \\ s_{2i} &= \sigma(\langle s_0, s_1, \ldots, s_{2i-1} \rangle) \\ s_{2i+1} &= \tau(\langle s_0, s_1, \ldots, s_{2i} \rangle) \end{aligned}$$

A play of the game determined by the strategies σ and τ 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*.*

Given two strategies $\sigma \in \text{Strat}^I(\Gamma)$ and $\tau \in \text{Strat}^{II}(\Gamma)$, we will often write $\sigma \star \tau$ for the play determined by these two strategies.

Definition 2.4 (Determinacy)**.** *Let* $\Gamma = (X, R, D, \Phi)$ *be a game and let* $\mathcal{S} = Strat^I(\Gamma)$ *and* $\mathcal{T} = Strat^{II}(\Gamma)$*. Then* Γ *is* determined *with* value v *if:*

$$glb_{\tau \in \mathcal{T}} lub_{\sigma \in \mathcal{S}} \Phi(\sigma \star \tau) = lub_{\sigma \in \mathcal{S}} glb_{\tau \in \mathcal{T}} \Phi(\sigma \star \tau) = v$$

The following lemma can be established (see for example [Myc92]):

Lemma 2.5. *Let $\Gamma = (X, R, D, \Phi)$ be a game and let $\mathcal{S} = Strat^{I}(\Gamma)$ and $\mathcal{T} = Strat^{II}(\Gamma)$. Then:*

$$glb_{\tau\in\mathcal{T}}\, lub_{\sigma\in\mathcal{S}}\, \Phi(\sigma \star \tau) \geq lub_{\sigma\in\mathcal{S}}\, glb_{\tau\in\mathcal{T}}\, \Phi(\sigma \star \tau)$$

2.2 A Formal Definition of the Game

Let $G = (\Sigma, N, P, S)$ be a Boolean grammar, let $A \in N$ and let $w \in \Sigma^*$. We will define the perfect information game $\Gamma_G(A, w) = (X, R_{(A,w)}, D, \Phi_{(A,w)})$.

Definition 2.6. *Let $u \in \Sigma^*$. Then, a* partition *π of u of length n, is a sequence $\langle u_1, \dots, u_n\rangle$ such that $u_i \in \Sigma^*, 1 \leq i \leq n$, and $u_1 \cdots u_n = u$. We denote by Π_n the set of all partitions of length n.*

The game $\Gamma_G(A, w)$ can now be formally defined. We first define the set of moves:

$$\begin{array}{rcl} X & = & \{(\alpha, w) \mid \alpha \in (\Sigma \cup N)^*, w \in \Sigma^*\} \cup \\ & & \{(\neg\alpha, w) \mid \alpha \in (\Sigma \cup N)^*, w \in \Sigma^*\} \cup \\ & & \{(R, w) \mid R \in P, w \in \Sigma^*\} \cup \\ & & \{(\alpha, \pi) \mid \alpha \in (\Sigma \cup N)^*, \pi \in \Pi_{|\alpha|}\} \cup \\ & & \{(\text{I've won}), (\text{I've lost})\} \end{array}$$

We next define the tree $R_{(A,w)}$ of the game $\Gamma_G(A, w)$: $R_{(A,w)}$ consists of all sequences $\langle x_0, \dots, x_{n-1}\rangle$, which satisfy the following restrictions for each $k < n - 1$:

R0. $x_0 = (A \to l_1 \& \cdots \& l_m, w)$, where $A \to l_1 \& \cdots \& l_m$ is a rule of G.

R1. If $x_k = (B \to l_1 \& \cdots \& l_m, u)$, then $x_{k+1} = (l_i, u)$, where $1 \leq i \leq m$.

R2. If $x_k = (B, u)$, where $B \in N$, then $x_{k+1} = (B \to l_1 \& \cdots \& l_m, u)$, where $B \to l_1 \& \cdots \& l_m$ is a rule of G.

R3. If $x_k = (\neg\beta, u)$, then $x_{k+1} = (\beta, u)$. A transition of this form from x_k to x_{k+1} will be called a *role-switch*.

R4. If $x_k = (\beta, u)$, where $\beta \in E$, then $x_{k+1} = (\beta, \pi)$, where π is a partition of u of length $|\beta|$.

R5. If $x_k = (\beta, \pi)$, then $x_{k+1} = (\beta(i), \pi(i))$, where $1 \leq i \leq |\beta|$.

R6. $x_k = (u, u)$, where $u \in \Sigma^*$, or $x_k = (\text{I've lost})$, then $x_{k+1} = (\text{I've won})$.

R7. $x_k = (v, u)$, where $v, u \in \Sigma^*$ and $v \neq u$, or $x_k = (\text{I've won})$, then $x_{k+1} = (\text{I've lost})$.

The strategies of the two players are as defined in Definition 2.2. Consider now the set of rewards. We define $D = \{0, \frac{1}{2}, 1\}$. In other words, a play of the game can be assigned the value 0 (Player II has won the play), the value 1 (Player I has won), or the value $\frac{1}{2}$ (the result is a tie).

Definition 2.7 (True-play, False-play)**.** *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar, $w \in \Sigma^*$ and $A \in N$, and let s be a play of the corresponding game $\Gamma_G(A, w)$. Then, s is called a* true-play *if either Player I plays the* (I've won) *move in s or s is a genuinely infinite play that contains an odd number of role-switches. Similarly, s is called a* false-play *if either Player I plays the* (I've lost) *move in s or s is a genuinely infinite play that contains an even number of role-switches.*

The payoff function is defined as follows:

$$\Phi_{(A,w)}(s) = \begin{cases} 1, & \text{if } s \text{ is a true-play} \\ 0, & \text{if } s \text{ is a false-play} \\ \frac{1}{2}, & \text{otherwise} \end{cases}$$

3 Equivalence to the Well-Founded Semantics

Theorem 3.1. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and M_G be its well-founded model. For every $A \in N$ and $w \in \Sigma^*$, the game $\Gamma_G(A, w)$ is determined with value $M_G(A)(w)$.*

The proof of the theorem is based on the following steps: we start by slightly extending the game to a more general context. We then define two optimal strategies for the two players of the new game. Finally, we use these two strategies in order to establish a more general statement which implies the above theorem as a special case. Because of space considerations, we omit the proof entirely.

4 Conclusions and Future Work

We have presented a novel semantics for Boolean grammars which has been inspired by techniques that have been developed in the logic programming domain. Under this new semantics every Boolean grammar has a distinguished (three-valued) model that satisfies its rules. Moreover, we have shown that this model is the least fixed-point of an appropriate operator that is associated with the grammar. Finally, we have demonstrated that every Boolean grammar can be transformed into an equivalent one in a binary normal form. For grammars in this normal form, we have derived an $\mathcal{O}(n^3)$ parsing algorithm.

We believe that the well-founded semantics will prove to be a useful tool for the further development of the theory of Boolean grammars. In particular, the well-founded approach has already been used in order to prove that the locally stratified construction is well-defined (see [NR08] for details). Also, it is expected that the well-founded semantics and its corresponding parsing algorithm can form the basis of general implementations of Boolean grammars. On the more theoretical side, the formal machinery behind the well-founded semantics can help to the further development of many-valued formal language theory (see for example [EK07]).

Furthermore, we have presented an infinite game semantics for Boolean grammars and have demonstrated that it is equivalent to the well-founded semantics of this type of grammars. The simplicity of the new semantics stems mainly from its anthropomorphic flavor. In this respect, it differs from the well-founded semantics whose construction requires a more heavy mathematical machinery. We believe that these two semantical approaches can be used in a complementary way in the study of Boolean grammars. In our opinion, the game-theoretic approach will prove useful in establishing the correctness of meaning-preserving transformations for Boolean grammars. On the other hand, the well-founded semantics appears to be more useful in computing the meaning of *specific* grammars. This is due to the iterative-inductive flavor of the well-founded approach (see [NR08] for an example of an iterative computation of the meaning of a Boolean grammar using a procedure that was inspired by the well-founded construction).

Closing, we would like to express our strong belief that a further investigation of the connections between formal language theory and the theory of logic programming will prove to be very rewarding.

References

[EK07] Z. Esik and W. Kuich. Boolean Fuzzy Sets. *International Journal of Foundations of Computer Science*, 18(6):1197–1207, 2007.

[Myc92] J. Mycielski. Games with perfect information. *Handbook of Game Theory*, pages 41–70, 1992.

[NR08] C. Nomikos and P. Rondogiannis. Locally stratified Boolean grammars. *Information and Computation*, 206(9-10):1219–1233, 2008.

[Okh04] A. Okhotin. Boolean grammars. *Information and Computation*, 194(1):19–48, 2004.

[PP90] H. Przymusinska and T. Przymusinski. Semantic issues in deductive databases and logic programs. *Formal Techniques in Artificial Intelligence*, 1990.