Chapter 4

Boolean grammars

4.1 Definition of Boolean grammars

Definition 4.1 (\[17\]). A Boolean grammar is a quadruple $G = (\Sigma, N, P, S)$, in which:

- $\Sigma$ and $N$ are disjoint finite nonempty sets of terminal and nonterminal symbols respectively;
- $P$ is a finite set of grammar rules, each of the form
  
  $$A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n$$
  
  (where $A \in N$, $m, n \geq 0$, $m + n \geq 1$ and $\alpha_i, \beta_j \in (\Sigma \cup N)^*$) (4.1)

- $S \in N$ is a nonterminal designated as the start symbol.

Intuitively, a rule (4.1) can be read as “if a string satisfies the syntactical conditions $\alpha_1, \ldots, \alpha_m$ and does not satisfy any of the syntactical conditions $\beta_1, \ldots, \beta_n$, then this string satisfies the condition represented by the nonterminal $A$”. This intuitive interpretation is sufficient to construct grammars, but although it is clear for “reasonably written” grammars, the use of negation can, in general, lead to logical contradictions. (consider the grammar $S \rightarrow \neg S$), and for that reason the task of defining a mathematically sound formal semantics for Boolean grammars is far from being trivial.

All existing definitions of Boolean grammars $[17, 10]$ start with representing a grammar as a system of language equations with concatenation, union, intersection and complementation.

Definition 4.2. For every Boolean grammar $G = (\Sigma, N, P, S)$, the associated system of language equations has the following equations:

$$A = \bigcup_{A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n \in P} \left[ \bigcap_{i=1}^{m} \alpha_i \cap \bigcap_{j=1}^{n} \neg \beta_j \right] \quad \text{(for all } A \in N) \quad (4.2)$$

Need not have solutions: $S \rightarrow \neg S$.

May have multiple pairwise incomparable solutions: $S \rightarrow \neg A$, $A \rightarrow A$. (hence no definition by least solutions).

In general, systems of language equations of the form (4.2) have a high expressive power and the associated undecidability results $[19]$.

4.1.1 Strongly unique solution

Even unique solutions are too much. The class of languages represented by their unique solutions is exactly the class of recursive languages. The way these languages are represented contradicts the intuitive semantics of Boolean grammars defined above.
Imposing some restrictions upon language equations leads to a feasible semantics for Boolean grammars. Here is one possible restriction (a condition stronger than solution uniqueness):

\[(\Sigma \leq \ell = \{ w \mid w \in \Sigma^*, |w| \geq \ell \})\]

**Definition 4.3.** Let \( G = (\Sigma, N, P, S) \) be a Boolean grammar, let \( (4.2) \) be the associated system of language equations. Suppose that for every number \( \ell \geq 0 \) there exists a unique vector of languages \( (\ldots, L_C, \ldots)_{C \in N} (L_C \subseteq \Sigma\leq\ell) \), such that a substitution of \( L_C \) for \( C \), for each \( C \in N \), turns every equation \( (4.2) \) into an equality modulo intersection with \( \Sigma\leq\ell \).

Then \( G \) complies to the semantics of a strongly unique solution, and, for every \( A \in N \), the language \( L_G(A) \) can be defined as \( L_A \) from the unique solution of this system. The language generated by the grammar is \( L(G) = L_G(S) \).

### 4.1.2 Well-founded fixed point

Semantics for Boolean grammars based upon three-valued languages, proposed by Koutouriotis, Nomikos and Rondogiannis [10].

Three-valued languages: mappings from \( \Sigma^* \) to \{0, \frac{1}{2}, 1\}. Equivalently: defined by pairs \( \{(L, L') \mid L \subseteq L' \subseteq \Sigma^*\} \). Here \( L \) and \( L' \) represent a lower bound and an upper bound on a language that is not known precisely. If \( L = L' \), the language is completely defined. A pair \( (\varnothing, \Sigma^*) \) means a language about which nothing is known. The set of such pairs is denoted by \( 3\Sigma^* \).

Operations on three-valued languages:

\[
(K, K') \cup (L, L') = (K \cup L, K' \cup L') \\
(K, K') \cap (L, L') = (K \cap L, K' \cap L') \\
(K, K')(L, L') = (KL, K'L')
\]

Two partial orderings on three-valued languages. Degree of truth: \( (K, K') \leq_T (L, L') \) if \( K \subseteq L \) and \( K' \subseteq L' \).

Degree of information: \( (K, K') \leq_I (L, L') \) if \( K \subseteq L \) and \( L' \subseteq K' \).

Both orderings are extended to vectors of three-valued languages. The truth-ordering has bottom element \( \varnothing_T = ((\varnothing, \varnothing), \ldots, (\varnothing, \varnothing)) \), that is, every language is completely defined as \( \varnothing \); the top element is \( ((\Sigma^*, \Sigma^*), \ldots, (\Sigma^*, \Sigma^*)) \). For the information-ordering, the bottom element is \( \bot_I = ((\varnothing, \Sigma^*), \ldots, (\varnothing, \Sigma^*)) \), that is, the languages are fully undefined. There is no top element for \( \leq_I \).

**Lemma 4.1.** Concatenation and all Boolean operations (including complementation) are monotone and continuous with respect to the information ordering. The same applies to every combination of these operations.

**Lemma 4.2.** Concatenation, union and intersection, as well as every combination thereof, are monotone and continuous with respect to the truth ordering.

**Definition 4.4.** Let \( G = (\Sigma, N, P, S) \) be a Boolean grammar, let \( N = \{A_1, \ldots, A_n\} \) and define a function \( \varphi : (3\Sigma^*)^n \rightarrow (3\Sigma^*)^n \) by

\[
[\varphi(L)]_A = \bigcup_{A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n \in P} \left[ \bigcap_{i=1}^m \alpha_i(L) \cap \bigcap_{j=1}^n \beta_j(L) \right] \quad \text{(for each } A \in N \text{)}
\]
The task is to associate a vector of three-valued languages $L_G$ to every grammar $G$, with the property that $L_G = \varphi(L_G)$.

The simplest way of constructing such a vector is by taking the least solution of the system $X = \varphi(X)$ with respect to $\sqsubseteq_I$. Since $\varphi$ is monotone and continuous with respect to this ordering, this least solution is given by the least upper bound

$$\bigcup_{k \geq 0} \varphi^k(\perp_I).$$

However, it does not meet the stated goals. In particular, it is not compatible with the context-free semantics: it assigns value $(\emptyset, \Sigma^*)$ to the grammar $S \to S$.

**Definition 4.5 (Well-founded semantics [10]).** Let $G = (\Sigma, N, P, S)$ be a Boolean grammar, let $N = \{A_1, \ldots, A_n\}$. Fix any vector of three-valued languages $K = ((L_1, L'_1), \ldots, (L_1, L'_1))$ and define a function $\Theta_K : (3^\Sigma)^n \to (3^\Sigma)^n$ by

$$[\Theta_K(L)]_A = \bigcup_{A \to \alpha_1 k \ldots \alpha_m k' \beta_1 \ldots \beta_n \in P} \left[ \bigcap_{i=1}^m [\alpha_i(L) \cap \bigcap_{j=1}^n [\beta_j(K) \setminus \perp_T] \right]$$

(for each $A \in N$)

Furthermore, define

$$\Omega(K) = \bigcup_{\ell \geq 0} \Theta^\ell_K(\perp_T).$$

and let

$$M = \bigcup_{k \geq 0} \Omega^k(\perp_I).$$

Then, according to the well-founded semantics of Boolean grammars, $L_G(A) = [M]_A$.

**Lemma 4.3.** For every $K$, the function $\Theta_K$ is monotone and continuous with respect to $\sqsubseteq_T$.

**Proof.** To see that $\Theta_K$ is monotone, consider that whenever negation is used in its definition, it is applied to an expression involving only $K$ but not $L$, which can therefore be regarded as a constant expression. As the rest of the expression uses only union, intersection and constant languages, $\Theta_K(L)$ is $\sqsubseteq_T$-monotone as a function of $L$.

It is $\sqsubseteq_T$-continuous as a combination of continuous operations. 

**Lemma 4.4.** The function $\Omega$ is monotone and continuous with respect to $\sqsubseteq_I$.

**Proof.** To see that $\Omega$ is $\sqsubseteq_I$-monotone, it has to be proved that whenever $K \sqsubseteq_I K'$ for $K, K' \in (3^\Sigma)^n$, it holds that $\Omega(K) \sqsubseteq_I \Omega(K')$, that is, that

$$\bigcup_{\ell \geq 0} \Theta^\ell_K(\perp_T) \sqsubseteq_I \bigcup_{\ell \geq 0} \Theta^\ell_{K'}(\perp_T)$$

Consider that $\Theta_K(L)$, viewed as a function of two vector arguments, $K$ and $L$, is $\sqsubseteq_I$-monotone. Now it is claimed that

$$\Theta^\ell_K(\perp_T) \sqsubseteq_I \Theta^\ell_{K'}(\perp_T),$$

and this shall be proved by induction on $\ell$. The basis, $\perp_T \sqsubseteq_I \perp_T$, holds true, and the induction step follows by the monotonicity of $\Theta(K, L)$, since [13] implies $\Theta_K(\Theta^\ell_K(\perp_T)) \sqsubseteq_I \Theta_{K'}(\Theta^\ell_{K'}(\perp_T))$. 

**Lemma 4.5.** $M = \varphi(M)$. 

Proof. Since $M$ is defined as the least upper bound
\[ M = \bigcup_{k \geq 0} \Omega^k(\bot_I), \]
and the function $\Omega$ is monotone and continuous with respect to $\sqsubseteq_I$ by Lemma 4.4, $M$ satisfies the following equation:
\[ M = \Omega(M). \]
Expanding the definition of $\Omega$, this means that
\[ M = \bigcup_{\ell \geq 0} \Theta_M^\ell(\bot_T). \]
Here the function $\Theta_M$ is monotone and continuous with respect to $\sqsubseteq_T$ by Lemma ??, and accordingly,
\[ M = \Theta_M(M). \]
As the function $\Theta_M(M)$ coincides with $\varphi$, the claim is proved.

Example 4.1. According to the well-founded semantics, the Boolean grammar
\[
S \rightarrow \neg A \\
A \rightarrow A
\]
has $L_G(S) = \Sigma^*$ and $L_G(A) = \varnothing$.

4.2 Examples

Example 4.2. The following Boolean grammar generates the language $\{a^m b^n c^n \mid m, n \geq 0, m \neq n\}$:
\[
S \rightarrow AB \& \neg DC \\
A \rightarrow aA \mid \varepsilon \\
B \rightarrow bBc \mid \varepsilon \\
C \rightarrow cC \mid \varepsilon \\
D \rightarrow aDb \mid \varepsilon
\]
The rules for the nonterminals $A$, $B$, $C$ and $D$ are context-free. Then the propositional connectives in the rule for $S$ specify the following combination of the conditions given by $AB$ and $DC$:
\[
\{a^m b^n c^n \mid m, n \geq 0, m \neq n\} = \{a^i b^j c^k \mid j = k \text{ and } i \neq j\} = L(AB) \cap \overline{L(DC)}
\]

Example 4.3. The following Boolean grammar generates the language $\{ww \mid w \in \{a, b\}^*\}$:
\[
S \rightarrow \neg AB \& \neg BA \& C \\
A \rightarrow XAX \mid a \\
B \rightarrow XBX \mid b \\
C \rightarrow XXC \mid \varepsilon \\
X \rightarrow a \mid b
\]
Again, according to the intuitive semantics, the nonterminals $A$, $B$, $C$ and $X$ should generate the appropriate context-free languages, and

\[
L(A) = \{uav \mid u, v \in \{a, b\}^*, |u| = |v|\}, \\
L(B) = \{ubv \mid u, v \in \{a, b\}^*, |u| = |v|\}.
\]

This implies

\[
L(AB) = \{uavxby \mid u, v, x, y \in \{a, b\}^*, |u| = |x|, |v| = |y|\},
\]

that is, $L(AB)$ contains all strings of even length with a mismatched $a$ on the left and $b$ on the right (in any position). Similarly,

\[
L(BA) = \{ubvxay \mid u, v, x, y \in \{a, b\}^*, |u| = |x|, |v| = |y|\}
\]

specifies the mismatch formed by $b$ on the left and $a$ on the right. Then the rule for $S$ specifies the set of strings of even length without such mismatches:

\[
L(S) = L(AB) \cap L(BA) \cap \{aa, ab, ba, bb\}^* = \{ww \mid w \in \{a, b\}^*\}.
\]

Example 4.4 (adapted from Leiss [12]). Consider the following Boolean grammar:

\[
S \rightarrow aAA \\
A \rightarrow ¬BB \\
B \rightarrow ¬CC \\
C \rightarrow ¬S
\]

It generates the language

\[
L(G) = \{a^n \mid \exists k \geq 0 : 2^{3k} \leq n < 2^{3k+2}\},
\]

while its nonterminals $A, B, C$ generate the following languages:

\[
L_G(A) = \{a^n \mid \exists k \geq 0 : 2^{3k} - 1 \leq n < 2^{3k+1}\}, \\
L_G(B) = \{a^n \mid \exists k \geq 0 : 2^{3k+1} \leq n < 2^{3k+2}\}, \text{ and} \\
L_G(C) = \{a^n \mid \exists k \geq 0 : 2^{3k+2} \leq n < 2^{3k+3}\} \cup \{\varepsilon\}.
\]

Note that this grammar transcribes the following one-variable language equation:

\[
X = aX^2 + 2^2.
\]

This equation was constructed by Leiss [12], who gave no explanation of how it works and how it was found. The solution can be verified by substitution. This is the first example of a language equation over a unary alphabet with a non-regular solution.

4.3 Binary normal form

Definition 4.6 (Binary normal form [17]). A Boolean grammar $G = (\Sigma, N, P, S)$ is in the binary normal form if every rule in $P$ is of the form

\[
A \rightarrow B_1C_1\ldots \& B_mC_m\& ¬D_1E_1\ldots \& ¬D_nE_n\& ¬\varepsilon \quad (m, n \geq 0, \ m + n \geq 1, \ B_i, C_i, D_j, E_j \in N) \\
A \rightarrow a \\
S \rightarrow \varepsilon \quad \text{(only if} \ S \text{ does not appear in right-hand sides of rules)}
\]
Theorem 4.1 ([17]). Every Boolean grammar can be effectively transformed to a Boolean grammar in the binary normal form generating the same language.

Lemma 4.6 ([17]). For every Boolean grammar $G = (\Sigma, N, P, S)$ compliant to the semantics of strongly unique solution, let

$$\lambda(s_1 \ldots s_\ell) = \{ s_{i_1} \ldots s_{i_k} \mid k \geq 1; 1 \leq i_1 < \ldots < i_k \leq \ell; i \notin \{i_1, \ldots, i_k\} \Rightarrow \varepsilon \in L_G(s_i)\}$$

Then the grammar $G' = (\Sigma, N, P', S)$ with the set of rules

$$A \rightarrow \alpha_1' \& \ldots \& \alpha_m' \& \neg \beta_1' \& \ldots \& \neg \beta_m'$$

$$\{\exists A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_m \in P : \alpha_i' \in \lambda(\alpha_i) \text{ for all } i, \{\beta_{j1}', \ldots, \beta_{j\ell}'\} = \lambda(\beta_j) \text{ for all } j\}$$

is compliant to the semantics of strongly unique solution and satisfies $L_G'(A) = L_G(A) \setminus \{\varepsilon\}$ for all $A \in N$, in particular $L(G') = L(G) \setminus \{\varepsilon\}$.

Now consider any Boolean grammar $G = (\Sigma, N, P, S)$ without $\varepsilon$ conjuncts. Let $N = \{A_1, \ldots, A_n\}$ and let $\hat{\Gamma} = \bigcup A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_m \{\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_m\} \setminus N$ be the set of all conjuncts except the unit conjuncts. For every set $\Gamma \subseteq \hat{\Gamma}$, define a Boolean function $\varphi_\Gamma : \mathbb{B}^n \rightarrow \mathbb{B}^n$ by

$$[\varphi_\Gamma(x)]_A = \bigvee_{A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_m \in P} \left[ \bigwedge_{i=1}^m \rho_\Gamma(\alpha_i) \land \bigwedge_{j=1}^m \neg \rho_\Gamma(\beta_j) \right], \text{ where}
$$

$$\rho_\Gamma(\alpha) = \begin{cases} 1, & \text{if } \alpha \in \Gamma \\ 0, & \text{if } \alpha \in \Gamma \setminus \Gamma \\ x_i, & \text{if } \alpha = A_i \end{cases}$$

Lemma 4.7 ([17]). Let $G = (\Sigma, N, P, S)$ be a Boolean grammar without $\varepsilon$ conjuncts compliant to the semantics of strongly unique solution. Then the grammar $G' = (\Sigma, N, P', S)$ with the set of rules

$$A \rightarrow \gamma_1 \& \ldots \& \gamma_m \& \neg \delta_1 \& \ldots \& \neg \delta_m \& \neg \varepsilon,$$

where

$$\Gamma = \{\gamma_1, \ldots, \gamma_m\}, \hat{\Gamma} \setminus \Gamma = \{\delta_1, \ldots, \delta_m\},$$

the system $x = \varphi_\Gamma(x)$ has a unique solution with $x_A = 1$

is compliant to the semantics of strongly unique solution and satisfies $L_G'(A) = L_G(A)$ for all $A \in N$. in particular $L(G') = L(G)$.

Every rule of the constructed grammar is typically as large as the original grammar. It represent one possible case of obtaining the unique solution modulo $M \cup \{w\}$ from the unique solution modulo $M$.

The rest of the transformation towards the binary normal form is straightforward.