Search Space Properties for Learning a Class of Constraint-based Grammars

Smaranda Muresan
School of Communication and Information
Rutgers University
4 Huntington St, New Brunswick, NJ, 08901
smuresan@rutgers.edu

Abstract

We discuss a class of constraint-based grammars, Lexicalized Well-Founded Grammars (LWFGs) and present the theoretical underpinnings for learning these grammars from a representative set of positive examples. Given several assumptions, we define the search space as a complete grammar lattice. In order to prove a learnability theorem, we give a general algorithm through which the top and bottom elements of the complete grammar lattice can be built.

1 Introduction

There has been significant interest in grammar induction on the part of both formal languages and natural language processing communities. In this paper, we discuss the learnability of a recently introduced constraint-based grammar formalism for deep linguistic processing, Lexicalized Well-Founded Grammars (LWFGs) (Muresan, 2006; Muresan and Rambow, 2007; Muresan, 2011). Most formalisms used for deep linguistic processing, such as Tree Adjoining Grammars (Joshi and Schabes, 1997) and Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) are not known to be accompanied by a formal guarantee of polynomial learnability. While stochastic grammar learning for statistical parsing for some of these grammars has been achieved using large annotated treebanks (e.g., (Hockenmaier and Steedman, 2002; Clark and Curran, 2007; Shen, 2006)), LWFG is suited to learning in *resource-poor settings*. LWFG's learning is a relational learning framework which characterizes the importance of substructures in the model not simply by frequency, as in most previous work, but rather linguistically, by defining a notion of *representative examples* that drives the acquisition process.

LWFGs can be seen as a type of Definite Clause Grammars (Pereira and Warren, 1980) where: 1) the Context-Free Grammar backbone is extended by introducing a *partial ordering relation* among *delexicalized* nonterminals (well-founded), 2) nonterminals are augmented with strings and their syntactic-semantic representations; and 3) grammar rules have two types of constraints: one for semantic composition and one for ontology-based semantic interpretation (Muresan, 2006). In LWFG every string w is associated with a syntactic-semantic representation called semantic molecule $\binom{h}{b}$. We call the tuple $(w, \binom{h}{b})$ a *syntagma*:

$$\text{(big table, } \left(\begin{array}{ll} h & \left[\begin{array}{ll} \text{cat} & \text{np} \\ \text{head} & X \\ \text{nr} & sg \end{array}\right] \\ b & \left\langle X_1.\text{isa} = \text{big}, X.Y = X_1, X.\text{isa=table} \right\rangle \end{array} \right) \text{)}$$

The language generated by a LWFG consists of syntagmas, and not strings.

There are several properties and assumptions that are essential for LWFG learnability: 1) partial ordering relation on the delexicalized nonterminal set (well-founded property); 2) each string has its linguistic category known (e.g., `np` for the phrase *big table*); 3) LWFGs are unambiguous; 4) LWFGs are non-terminally separable (Clark, 2006). Regarding unambiguity, we need to emphasize that unambiguity is relative to a set of syntagmas (pairs of strings and their syntactic-semantic representations) and not to a set of natural language strings. For example, the sentence *I saw the man with a telescope* is ambiguous at the string level (PP-attachment ambiguity), but it is unambiguous if we consider the syntagmas associated with it.

Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11), pages 171–179,
Paris, September 2012.

In this paper, for clarity of presentation, we make abstraction of the semantic representation and grammar constraints, and discuss the theoretical underpinnings of Well-Founded Grammars (WFGs), which have all the properties of LWFGs that assures polynomial learnability. By defining the operational and denotational semantics of WFGs, we are able to formally define the representative set of WFGs. Giving several assumptions, we define the search space for WFG learning as a complete grammar lattice by defining the least upper bound and the greatest lower bound operators. The grammar lattice preserves the parsing of the representative set. We give a theorem showing that this lattice is a complete grammar lattice. In order to give a learnability theorem, we give a general algorithm through which the top and the bottom elements of the complete grammar lattice can be built. The theoretical results obtained in this paper hold for the LWFG formalism. This theoretical result proves that the practical algorithms introduced by Muresan (2011) converge to the same target grammar.

2 Well-Founded Grammars

Well-Founded Grammars are a subclass of Context-Free Grammars where there is a partial ordering relation on the set of non-terminals.

Definition 1. *A Well-Founded Grammar (WFG) is a 5-tuple, $G = \langle \Sigma, N_G, \succeq, P, S \rangle$ where:*

1. *Σ is a finite set of terminal symbols.*
2. *N_G is a finite set of nonterminal symbols, where $N_G \cap \Sigma = \emptyset$.*
3. *$\succeq$ is a partial ordering relation on the set of nonterminals N_G*
4. *P is the set of grammar rules, $P = P_\Sigma \cup P_G$, $P_\Sigma \cap P_G = \emptyset$, where:*
   a) *P_Σ is the set of grammar rules whose right-hand side are terminals, $A \rightarrow w$, where $A \in N_G$ and $w \in \Sigma$ (empty string cannot be derived). We denote $pre(N_G) \subseteq N_G$ the set of pre-terminals, $pre(N_G) = \{A | A \in N_G, w \in \Sigma, A \rightarrow w \in P_\Sigma\}$.*
   b) *P_G is the set of grammar rules $A \rightarrow B_1 \dots B_n$, where $A \in (N_G - pre(N_G))$, $B_i \in N_G$. For brevity, we denote a rule by $A \rightarrow \beta$, where $A \in (N_G - pre(N_G)), \beta \in N_G^+$. For every grammar rule $A \rightarrow \beta \in P_G$ there is a direct relation between the left-hand side nonterminal A and all the nonterminals on the right-hand side $B_i \in \beta$ (i.e., $A \succeq B_i$, or $B_i \succeq A$). If for all $B_i \in \beta$ we have that $A \succeq B_i$ and $A \neq B_i$, the grammar rule $A \rightarrow \beta$ is an ordered non-recursive rule. Each nonterminal symbol $A \in (N_G - pre(N_G))$ is a left-hand side in at least one ordered non-recursive rule. In addition, the empty string cannot be derived from any nonterminal symbol and cycles are not allowed.*
5. *$S \in N_G$ is the start nonterminal symbol, and $\forall A \in N_G, S \succeq A$ (we use the same notation for the reflexive, transitive closure of $\succeq$).*

$\Sigma = \{0,1,2,3,4,5,6,7,8,9,+,-,*,\div,(,)\}$
$N_G = \{D, Sum, Prod, Lbr, Rbr, N, F, T, E\}$
$pre(N_G) = \{D, Sum, Prod, Lbr, Rbr\}$

P_Σ	P_G
$D \rightarrow 0\|1\|2\|3\|4\|5\|6\|7\|8\|9$	$N \rightarrow D$
$Sum \rightarrow +\|-$	$N \rightarrow N\ D$
$Prod \rightarrow *\|\div$	$F \rightarrow N$
$Lbr \rightarrow ($	$F \rightarrow Lbr\ E\ Rbr$
$Rbr \rightarrow)$	$T \rightarrow F$
	$T \rightarrow T\ Prod\ F$
	$E \rightarrow T$
	$E \rightarrow E\ Sum\ T$

Figure 1: A WFG for Mathematical Expressions

Besides the partial ordering relations $\succeq$, in WFGs the set of production rules P is split in P_Σ and P_G. For learning, P_Σ is given, while the grammar rules in P_G are learned.

In Figure 1 we give a WFG for Mathematical Expressions that will be used as a simple illustrative example to present the formalism and the foundation of the search space for WFG learning.

Every CFG $G = \langle \Sigma, N_G, P_\Sigma \cup P_G, S \rangle$ can be efficiently tested to see whether it is a Well-Founded Grammar.

The derivation in WFGs is called *ground derivation* and it can be seen as the bottom up counterpart of the usual derivation. Given a WFG, G, the *ground derivation* relation, $\overset{*G}{\Rightarrow}$, is defined as: $\frac{A \rightarrow w}{A \overset{*G}{\Rightarrow} w}$ $(A \rightarrow w \in P_\Sigma)$, and $\frac{B_i \overset{*G}{\Rightarrow} w_i,\ i=1,\dots,n \quad A \rightarrow B_1 \dots B_n}{A \overset{*G}{\Rightarrow} w \quad w = w_1 \dots w_n}$ $(w \in \Sigma^+)$.

The language of a grammar G is the set of all strings generated from the start symbol S, i.e., $L(G) = \{w | w \in \Sigma^+, S \overset{*G}{\Rightarrow} w\}$. The *set of all strings* generated by *a grammar* G is $L_w(G) = \{w | w \in \Sigma^+, \exists A \in N_G, A \overset{*G}{\Rightarrow} w\}$. Extending the

notation, the set of strings generated by *a nonterminal* A of a grammar G is $L_w(A) = \{w | w \in \Sigma^+, A \in N_G, A \overset{*G}{\Rightarrow} w\}$, and the set of strings generated by *a rule* $A \rightarrow \beta$ of a grammar G is $L_w(A \rightarrow \beta) = \{w | w \in \Sigma^+, (A \rightarrow \beta) \overset{*G}{\Rightarrow} w\}$, where $(A \rightarrow \beta) \overset{*G}{\Rightarrow} w$ denotes the ground derivation $A \overset{*G}{\Rightarrow} w$ obtained using the rule $A \rightarrow \beta$ in the last derivation step. For a WFG G, we call a set of substrings $E_w \subseteq L_w(G)$ a sublanguage of G.

Operational Semantics of WFGs. It has been shown that the operational semantics of a CFG corresponds to the language of the grammar (Wintner, 1999). Analogously, the operational semantics of a WFG, G, is the set of all strings generated by the grammar, $L_w(G)$.

Denotational Semantics of WFGs. As discussed in literature (Pereira and Shieber, 1984; Wintner, 1999), the denotational semantics of a grammar is defined through a fixpoint of a transformational operator associated with the grammar. Let $I \subseteq L_w(G)$ be a subset of all strings generated by a grammar G. We define the *immediate derivation operator* T_G: $2^{L_w(G)} \rightarrow 2^{L_w(G)}$, s.t.: $T_G(I) = \{w \in L_w(G) |$ if $(A \rightarrow B_1 \dots B_n) \in P_G \wedge B_i \overset{*G}{\Rightarrow} w_i \wedge w_i \in I$ then $A \overset{*G}{\Rightarrow} w\}$. If we denote $T_G \uparrow 0 = \emptyset$ and $T_G \uparrow (i+1) = T_G(T_G \uparrow i)$, then we have that for $i = 1, T_G \uparrow 1 = T_G(\emptyset) = \{w \in L_w(G) | A \in pre(N_G), A \overset{*G}{\Rightarrow} w\}$.This corresponds to the strings derived from preterminals, i.e., $w \in \Sigma$. T_G is analogous with the immediate consequence operator of definite logic programs (i.e., no negation) (van Emden and Kowalski, 1976; Denecker et al., 2001). T_G is monotonous and hence the least fixpoint always exists (Tarski, 1955). This least fixpoint is unique, as for definite logic programs (van Emden and Kowalski, 1976). We have $lfp(T_G) = T_G \uparrow \omega$, where ω is the minimum limit ordinal. Thus, the denotational semantics of a grammar G can be seen as the least fixpoint of the immediate derivation operator. An assumption for learning WFGs is that the rules corresponding to grammar preterminals, $A \rightarrow w \in P_\Sigma$, are given, i.e., $T_G(\emptyset)$ is given.

As in the case of definite logic programs, the denotational semantics is equivalent with the operational one, i.e., $L_w(G) = lfp(T_G)$. Based on T_G we can define the *ground derivation length (gdl)* for strings and the *minimum ground derivation length (mgdl)* for grammar rules, which are key concepts in defining the representative set E_R of a WFG.

$$gdl(w) = \min_{w \in T_G \uparrow i} (i)$$

$$mgdl(A \rightarrow \beta) = \min_{w \in L_w(A \rightarrow \beta)} (gdl(w))$$

2.1 Properties and Principles for WFG Learning

In this section we present the main properties of WFGs, discussing their importance for learning.

1) Partial ordering relation $\succeq$ (well-founded). In WFGs, the partial ordering relation $\succeq$ on the nonterminal set N_G allows the total ordering of grammar nonterminals and grammar rules, which allows the bottom-up learning of WFGs. WFG rules can be ordered or non-ordered, and they can be recursive or non-recursive. In addition, from the definition of WFGs, every non-terminal is a left-hand side in at least one ordered non-recursive rule, cycles are not allowed and the empty string cannot be derived, properties that guarantee the termination condition for learning.

2) Category Principle. Wintner (Wintner, 1999) calls observables for a grammar G all the derivable strings paired with the non-terminal that derives them: $Ob(G) = \{\langle w, A\rangle | w \in \Sigma, A \in N_G, A \overset{*G}{\Rightarrow} w\}$, i.e., $w \in L_w(A)$. We call w a *constituent* and A its *category*. For example, we can say that *1+1* is a constituent having the category *expression (E)*, *1*1* is a constituent having the category *term (T)*, and *1* is a constituent having the categories *digit(D), number (N), factor (F), term (T), expression (E)*. The Category Principle for WFGs states that *the observables are known a-priori*. When learning WFGs, the input to the learner are observables $\langle w, A\rangle$. The category is used by the learner as the name of the left-hand side (lhs) nonterminal of the learned grammar rule. The Category Principle is met for natural language, where observables (i.e., constituents and their linguistic categories) ca be identified: e.g., $\langle$*formal proposal, NP*$\rangle$, $\langle$*very loud, ADJP*$\rangle$.

3) Representative Set of a WFG (E_R). Given an unambiguous[1] Well-Founded Grammar G, a set of observables E_R is called a **representative set** of G iff for each rule $(A \rightarrow \beta) \in P_G$ there is a unique observable $\langle w, A\rangle \in E_R$ s.t.

[1] A WFG G is unambiguous if every string in $L(G)$ has only one derivation.

ID	E_R	$E_{gen} - E_R$
1	$\langle 1, N\rangle$	
2	$\langle 11, N\rangle$	$\langle 111, N\rangle$
3	$\langle 1, F\rangle$	$\langle 11, F\rangle$
4	$\langle (1), F\rangle$	$\langle (11), F\rangle\ \langle ((1)), F\rangle\ \langle (1 * 1), F\rangle\ \langle (1 + 1), F\rangle$
5	$\langle 1, T\rangle$	$\langle 11, T\rangle\ \langle (1), T\rangle$
6	$\langle 1 * 1, T\rangle$	$\langle 1 * 11, T\rangle\ \langle 11 * 1, T\rangle\ \langle 1 * (1), T\rangle\ \langle (1) * 1, T\rangle\ \langle 1 * 1 * 1, T\rangle$
7	$\langle 1, E\rangle$	$\langle 11, E\rangle\ \langle (1), E\rangle\ \langle 1 * 1, E\rangle$
8	$\langle 1 + 1, E\rangle$	$\langle 11 + 1, E\rangle\ \langle 1 + 11, E\rangle\ \langle (1) + 1, E\rangle\ \langle 1 + (1), E\rangle\ \langle 1 * 1 + 1, E\rangle, \langle 1 + 1 * 1, E\rangle\ \langle 1 + 1 + 1, E\rangle$

Figure 2: Examples for Learning the WFG in Figure 1

$gdl(w) = mgdl(A \rightarrow \beta)$, where $w \in L_w(G)$. The nonterminal A is the category of the string w. E_R contains the most simple strings ground derived by the grammar G paired with their categories. From this definition it is straightforward to show that $|E_R| = |P_G|$. The partial ordering relation on the nonterminal set induces a total order on the representative set E_R as well as on the set of grammar rules P_G.

For the WFG induction, the representative set E_R will be used by the learner to generate hypotheses (i.e., grammar rules). The category will give the name of the left-hand side nonterminals (lhs) of the learned grammar rules. An example of a representative set E_R for the mathematical expressions grammar from Figure 1 is given in Figure 2. For generalization the learner will use a generalization set of observables $E_{gen} = \{\langle w, A\rangle | A \in N_G \wedge A \overset{*G}{\Rightarrow} w\}$, where $E_R \subseteq E_{gen}$. An example of a generalization set E_{gen} for learning the WFG from Figure 1 is given in Figure 2.

4) Semantics of a WFG reduced to a generalization set E_{gen}. Given a WFG G and a generalization set E_{gen} (not necessarily of G) the set $\mathbb{S}(G) = \{\langle w, A\rangle | \langle w, A\rangle \in E_{gen} \wedge A \overset{*G}{\Rightarrow} w\}$ is called the semantics of G reduced to the generalization set E_{gen}. In other words, $\mathbb{S}(G)$ will contain all the pairs $\langle w, A\rangle$ in the generalization set whose strings w can be ground-derived by the grammar G, $w \in L_w(G)$. Given a grammar rule $r_A \in P_G$, we call $\mathbb{S}(r_A) = \{\langle w, A\rangle | \text{lhs}(r_A){=}A \wedge \langle w, A\rangle \in E_{gen} \wedge r_A \overset{*G}{\Rightarrow} w\}$ the semantics of r_A reduced to E_{gen}. The cardinality of $\mathbb{S}$ is used during learning as performance criterion.

5) E_R-parsing-preserving. We present the *rule specialization step* and the *rule generalization step* of unambiguous WFGs, such that they are *E_R-parsing-preserving* and are the inverse of each other. The property of E_R-parsing-preserving means that both the initial and the specialized/generalized rules ground-derive the same string w of the observable $\langle w, A\rangle \in E_R$. The **rule specialization step**:

$$\frac{A \rightarrow \alpha B \gamma \qquad B \rightarrow \beta}{A \rightarrow \alpha\beta\gamma}$$

is **E_R-parsing-preserving**, if there exists $\langle w, A\rangle \in E_R$ and $r_g \overset{*G}{\Rightarrow} w$ and $r_s \overset{*G'}{\Rightarrow} w$, where $r_g = A \rightarrow \alpha B\gamma$, $r_B = B \rightarrow \beta$, $r_s = A \rightarrow \alpha\beta\gamma$ and $r_g \in P_G$, $r_B \in P_G \cap P_{G'}$, $r_s \in P_{G'}$. We write $r_g \overset{r_B}{\vdash} r_s$.[2] The rule generalization step, which is also E_R-parsing-preserving, is defined as the inverse of the rule specialization step and denoted by $r_s \overset{r_B}{\dashv} r_g$.

Since $\langle w, A\rangle$ is an element of the representative set, w is derived in the minimum number of derivation steps, and thus the rule r_B is always an ordered, non-recursive rule. Examples of E_R-parsing-preserving rule specialization steps are given in Figure 3, where all rules derive the same representative example *1+1*. In the derivation step $E \rightarrow N\ Sum\ D \overset{N \rightarrow D}{\vdash} E \rightarrow D\ Sum\ D$ the ordered, non recursive rule $r_B = N \rightarrow D$ is used. If the recursive rule $N \rightarrow N\ D$ were used, we would obtain a specialized rule $E \rightarrow N\ D\ Sum\ D$ which does not preserve the parsing of the representative example *1+1*.

From both the specialization and the generalization step we have that $L_w(r_g) \supseteq L_w(r_s)$.

The goal of the rule specialization step is to obtain a new target grammar G' from G by specializing a rule of G ($G \overset{r}{\vdash} G'$). Extending the notation to allow for the transitive closure of rule specialization, we have that $G \overset{*}{\vdash} G'$, and we say that the grammar G' is **specialized** from the grammar G, using a finite number of rule specialization steps that are *E_R-parsing-preserving*. Similarly, the goal of the rule generalization step is to

[2] Grammar G is non-terminally separable (Clark, 2006).

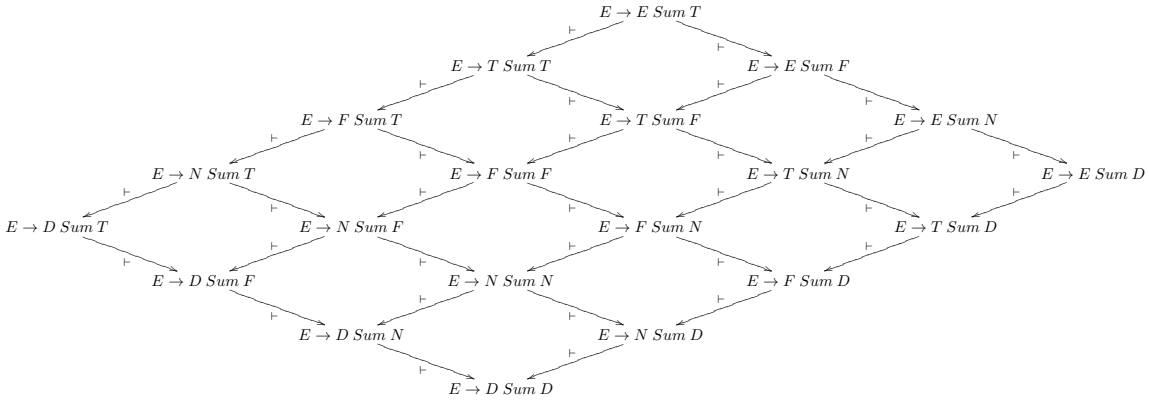


Figure 3: E_R-parsing preserving rule specialization steps for the grammar in Figure 1.

obtain a new target grammar G from G' by generalizing a rule of G'. Extending the notation to allow for transitive closure of rule generalization, we have that $G' \overset{*}{\dashv} G$, and we say that the grammar G is **generalized** from the grammar G' using a finite number of rule generalization steps that are *E_R-parsing-preserving*. That is, E_R is a representative set for both G and G'. The E_R-parsing-preserving property allows us to define a class of grammars that form a complete grammar lattice used as search space for WFGs induction, as detailed in the next section.

6) Conformal Property. A WFG G is called *normalized* w.r.t. a generalization set E_{gen}, if none of the grammar rules r_s of G can be further generalized to a rule r_g by the rule generalization step such that $\mathbb{S}(r_s) \subset \mathbb{S}(r_g)$. A WFG G is **conformal w.r.t. a generalization set** E_{gen} iff $\forall \langle w, A\rangle \in E_{gen}$ we have that $w \in L_w(G)$, and G is unambiguous and normalized w.r.t. E_{gen} and the rule specialization step guarantees that $\mathbb{S}(r_g) \supset \mathbb{S}(r_s)$ for all grammars specialized from G. This property allows learning only from positive examples.

7) Chains. We define a *chain* as a set of ordered unary branching rules: $\{B_k \to B_{k-1}, \ldots, B_2 \to B_1, B_1 \to \beta\}$ such that all these rules ground-derive the same string $w \in \Sigma^+$ (i.e., $B_k \succ B_{k-1} \cdots \succ B_1$ and $B_0 = \beta$, such that $B_i \overset{*G}{\Rightarrow} w$ for $0 \leq i \leq k$, where $B_i \in N_G$ for $1 \leq i \leq k$). For our grammar $\{E \to T, T \to F, F \to N, N \to D\}$ is a chain, all these rules ground-deriving the string *1*. Chains are used to generalize grammar rules during WFG learning (Fig. 3).

All the above mentioned properties are used to define the search space of WFG learning as a complete grammar lattice.

2.2 Grammar Lattice as Search Space

In this section we formally define a grammar lattice $\mathfrak{L} = \langle \mathcal{L}, \sqsupseteq \rangle$ that will be the search space for WFG learning. We first define the set of lattice elements $\mathcal{L}$.

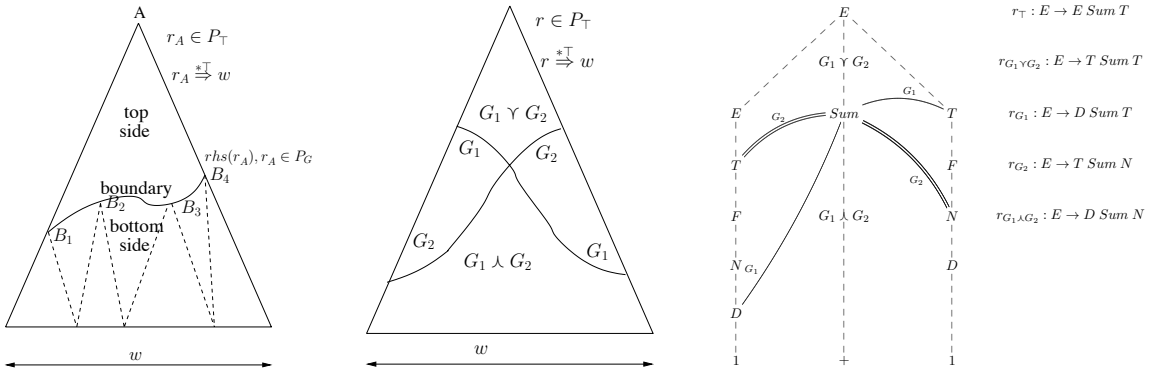
Let $\top$ be a WFG conformal to a generalization set E_{gen} that includes the representative set E_R of the grammar $\top$ ($E_{gen} \supseteq E_R$). Let $\mathcal{L} = \{G | \top \overset{*}{\vdash} G\}$ be the set of grammars specialized from $\top$. We call $\top$ the *top element* of $\mathcal{L}$, and $\bot$ the *bottom element* of $\mathcal{L}$, if $\forall G \in \mathcal{L}, \top \overset{*}{\vdash} G \wedge G \overset{*}{\vdash} \bot$. The bottom element, $\bot$, is the grammar specialized from $\top$, such that the right-hand side of all grammar rules contains only preterminals. We have $\mathbb{S}(\top) = E_{gen}$ and $\mathbb{S}(\bot) \supseteq E_R$.

There is a partial ordering among the elements of $\mathcal{L}$ (the subsumption $\sqsupseteq$), which we define below.

Definition 2. *If $G, G' \in \mathcal{L}$, we say that G subsumes G', $G \sqsupseteq G'$, iff $G \overset{*}{\vdash} G'$ (i.e., G' is specialized from G, and G is generalized from G')*

We have that for $G, G' \in \mathcal{L}$, if $G \sqsupseteq G'$ then $\mathbb{S}(G) \supseteq \mathbb{S}(G')$. That means that the subsumption relation is semantic based.

In sum, the set $\mathcal{L}$ contains the grammars specialized from $\top$, while the binary subsumption relation $\sqsupseteq$ establishes a partial ordering in $\mathcal{L}$. The top element of the lattice $\top$ is a normalized WFG, the bottom element $\bot$ is a grammar specialized from $\top$, whose rules' right-hand sides consist of preterminals, and all the other lattice elements are WFGs that preserve the parsing of the representative set. In Figure 3, the rule $(E \to E\ Sum\ T) \in P_\top$, the rule $(E \to D\ Sum\ D) \in P_\bot$ and all rules ground-derive the representative string *1+1*.



(a) Boundary (b) lub and glb operators (c) Example for lub and glb

Figure 4

In order for $\mathfrak{L} = \langle \mathcal{L}, \sqsupseteq \rangle$ to form a lattice, we must define two operators: the *least upper bound* (lub), $\curlyvee$ and the *greatest lower bound* (glb), $\curlywedge$, such that for any two elements $G_1, G_2 \in \mathcal{L}$, the elements $G_1 \curlyvee G_2 \in \mathcal{L}$ and $G_1 \curlywedge G_2 \in \mathcal{L}$ exist (Tarski, 1955).

We first introduce the concept of boundary. Let $r_A \in P_\top$ be a rule in grammar $\top$, and r'_A its specialized rule in grammar G ($\top \sqsupseteq G$) (see Figure 4a). Let $pt(r_A \overset{*\top}{\Rightarrow} w)$ be the parse tree corresponding to the ground-derivation $r_A \overset{*\top}{\Rightarrow} w$. We call **boundary** of a grammar $G \in \mathcal{L}$ relative to $pt(r_A \overset{*\top}{\Rightarrow} w)$,[3] the right-hand side of the corresponding rule $r'_A \in P_G, r'_A \overset{*G}{\Rightarrow} w$, i.e. $bd(G) = \{B | r'_A \in P_G \wedge B \in rhs(r'_A)\}$[4](see Figure 4a). For the example in Figure 4c, the rule in grammar $\top$ is $E \to E\ Sum\ T$, the rule in grammar G_1 is $E \to D\ Sum\ T$. Thus, $bd(G_1) = \{D, Sum, T\}$ We define the **bottom-side** $bs(G)$ of a grammar G relative to the parse tree $pt(r_A \overset{*\top}{\Rightarrow} w)$, as the forest composed of all the subtrees in $pt(r_A \overset{*\top}{\Rightarrow} w)$ whose roots are on $bd(G)$ (e.g., subtrees with roots at B_1, B_2, B_3, B_4 in Figure 4a). For the example in Figure 4c, $bs(G_1)$ will be the forest of subtrees of the parse tree $pt(r_\top \overset{*\top}{\Rightarrow} 1+1)$ with the roots D, Sum and T on the boundary $bd(G_1)$ of grammar G_1. We define the **top-side** $ts(G)$ of a grammar G relative to the parse tree $pt(r_A \overset{*\top}{\Rightarrow} w)$, as the subtree in $pt(r_A \overset{*\top}{\Rightarrow} w)$ rooted at A and whose leaf nodes are on $bd(G)$ (e.g., B_1, B_2, B_3 and B_4 in Figure 4a). For the example in Figure 4c, $ts(G_1)$ will be the subtree of the parse tree $pt(r_\top \overset{*\top}{\Rightarrow} 1+1)$ rooted at E and the leaf nodes D, Sum and T on the boundary $bd(G_1)$ of the grammar G_1. We have that $ts(G) \cap bs(G) = bd(G)$, $ts(G) \cup bs(G) = pt(r_A \overset{*\top}{\Rightarrow} w)$.

For any two elements $G_1, G_2 \in \mathcal{L}$, the lub element of G_1, G_2 is the minimum element that has the boundary above the boundaries of G_1 and G_2. The glb element of G_1, G_2 is the maximum element that has the boundary below the boundaries of G_1 and G_2. Thus, lub and glb are defined such that for all grammar rules we have:

$$\begin{aligned} ts(G_1 \curlyvee G_2) &= ts(G_1) \cap ts(G_2) \\ bs(G_1 \curlywedge G_2) &= bs(G_1) \cap bs(G_2) \end{aligned} \tag{1}$$

as can be seen in Figure 4b, 4c.[5] In order to have a complete lattice, the property must hold $\forall \mathcal{G} \subseteq \mathcal{L}$:

$$\begin{aligned} ts(\curlyvee_{G \in \mathcal{G}} G) &= \bigcap_{G \in \mathcal{G}} ts(G) \\ bs(\curlywedge_{G \in \mathcal{G}} G) &= \bigcap_{G \in \mathcal{G}} bs(G) \end{aligned} \tag{2}$$

Lemma 1. *$\mathfrak{L} = \langle \mathcal{L}, \sqsupseteq \rangle$ together with the lub and glb operators guarantees that for any two grammars $G_1, G_2 \in \mathcal{L}$ the following property holds:$G_1 \curlyvee G_2 \sqsupseteq G_1, G_2 \sqsupseteq G_1 \curlywedge G_2$*

Theorem 1. *$\mathfrak{L} = \langle \mathcal{L}, \sqsupseteq \rangle$ together with the lub and glb operators forms a complete lattice.*

Proof. Besides the property given in Lemma 1, lub and glb operators are computed w.r.t. (2), such that we have $ts(\curlyvee_{G \in \mathcal{L}} G) = \bigcap_{G \in \mathcal{L}} ts(G) =$

[3] All grammars $G \in \mathcal{L}$ are E_R-parsing-preserving and all boundaries of G are in the parse trees of the ground derivations of $\top$ grammar rules.

[4] The notation of $bd(G), ts(G), bs(G)$ ignores the rule relative to which these concepts are defined, and in the remainder of this paper we implicitly understand that the relations hold for all grammar rules.

[5] The intersection of two trees is the maximum common subtree of those two trees, and similarly for forests of trees.

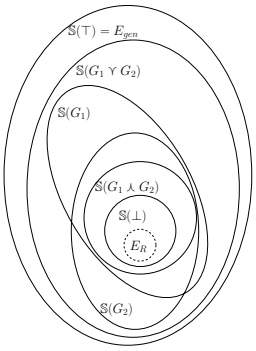


Figure 5: WFG semantics reduced to E_{gen}

$ts(\top)$, $bs(\curlywedge_{G\in\mathcal{L}} G) = \bigcap_{G\in\mathcal{L}} bs(G) = bs(\bot)$, which gives the uniqueness of $\top$ and $\bot$ elements. □

Similar to the subsumption relation $\sqsupseteq$, the lub $\curlyvee$ and glb $\curlywedge$ operators are semantic-based. In the complete lattice $\mathfrak{L} = \langle \mathcal{L}, \sqsupseteq \rangle$, $\forall G_1, G_2 \in \mathcal{L}$ we have:

$$\begin{aligned} &\mathbb{S}(G_1 \curlyvee G_2) \supseteq \mathbb{S}(G_1) \cup \mathbb{S}(G_2). \\ &\mathbb{S}(G_1 \curlywedge G_2) \subseteq \mathbb{S}(G_1) \cap \mathbb{S}(G_2) \end{aligned} \tag{3}$$

Thus, the complete grammar lattice is semantic-based (Figure 5). It is straightforward to prove that $\mathfrak{L} = \langle \mathcal{L}, \sqsupseteq \rangle$ has all the known properties (i.e., idempotency, commutativity, associativity, absorption, $\top$ and $\bot$ laws, distributivity).

2.3 Learnability Theorem

In oder to give a learnability theorem we need to show that $\bot$ and $\top$ elements of the lattice can be built. Through Algorithm 1 we show that giving the set of examples E_R and E_{gen}, the $\bot$ grammar can be built and the $\top$ grammar can be learned by generalizing the $\bot$ grammar. The grammar generalization is determinate if the rule generalization step is determinate. Before describing Algorithm 1 we introduce the concepts of *determinate generalizable* and give a lemma that states that given a grammar $\top$ conformal with E_{gen}, for any grammars G specialized from $\top$, all the grammar rules are determinate generalizable if all the chains of the $\top$ grammar are known.

Definition 3. *A grammar rule $r'_A \in P_G$ is determinate generalizable if for $\beta \in rhs(r'_A)$ there exists a unique rule $r_B = (B \rightarrow \beta)$ such that $r'_A \overset{r_B}{\dashv} r_A$ with $\mathbb{S}(r'_A) \subset \mathbb{S}(r_A)$. We use the notation $r'_A \overset{1\subset}{\dashv} r_A$ for the determinate generalization step with semantic increase.*

The only rule generalization steps allowed in the grammar induction process are those which guarantee the relation $\mathbb{S}(r_s) \subset \mathbb{S}(r_g)$ that ensures that all the generalized grammars belong to the grammar lattice. This property allows the grammar induction based only on positive examples. We use the notation $r'_A \overset{r_B\subset}{\dashv} r_A$ for the generalization step with semantic increase.

This step can be nondeterminate due to chain rules. Let $ch_\top$ be a chain of rules in a WFG $\top$ conformal w.r.t a generalization set E_{gen}, $ch_\top = \{B_k \rightarrow B_{k-1}, \ldots, B_2 \rightarrow B_1, B_1 \rightarrow \beta\}$. All the chain rules, but the last, are unary branching rules. The last rule is the minimal chain rule. For our example, $ch_\top = \{E \rightarrow T, T \rightarrow F, F \rightarrow N, N \rightarrow D\}$. For the $\bot$ grammar of a lattice that has $\top$ as its top element, the aforementioned chain becomes $ch_\bot = \{B_k \rightarrow \beta_\bot, \ldots, B_2 \rightarrow \beta_\bot, B_1 \rightarrow \beta_\bot\}$, where $\beta_\bot$ contains only preterminals and the rule order is unknown. By the E_R-parsing-preserving property of the rule specialization step, the same string is ground-derived from the $ch_\bot$ rules. Thus, the $\bot$ grammar is ambiguous. For our example, $ch_\bot = \{E \rightarrow D, T \rightarrow D, F \rightarrow D, N \rightarrow D\}$.

We denote by $ch = \{r_k, \ldots, r_2, r_1\}$, one or more chains in any lattice grammar, where the rule order is unknown. The minimal chain rules r_m can always be determined if $r_m \in ch$ s.t. $\forall r \in ch - \{r_m\} \wedge r_m \overset{r}{\dashv} r_{mg}$ we have that $\mathbb{S}(r_m) = \mathbb{S}(r_{mg})$ (see also `MinRule` algorithm). By the consequence of the conformal property, the generalization step $r_m \overset{r}{\dashv} r_{mg}$ is not allowed, since it does not produce any increase in rule semantics. That is, a minimal chain rule cannot be generalized by any other chain rule with an increase in its semantics. Given $ch_\bot$ and the aforementioned property of the minimal chain rules we can recover $ch_\top$ by `Chains_Recovery` algorithm.

Lemma 2. *Given a WFG $\top$ conformal w.r.t a generalization set E_{gen}, for any grammar G derived from $\top$ all rules are determinate generalizable if all chains of the grammar $\top$ (i.e., all $ch_\top$) are known (i.e., recovered by `Chains_Recovery` algorithm).*

Proof. The only case of rule generalization step nondeterminism with semantic increase is introduced by the derivation of the unary branching rules of ordered $ch_\top$, which yields the unordered

ID	$P_\perp$	$P_\top$
1	$N \rightarrow D$	$N \rightarrow D$
2	$N \rightarrow D\ D$	$N \rightarrow N\ D$
3	$F \rightarrow D$	$F \rightarrow N$
4	$F \rightarrow Lbr\ D\ Rbr$	$F \rightarrow Lbr\ E\ Rbr$
5	$T \rightarrow D$	$T \rightarrow F$
6	$T \rightarrow D\ Prod\ D$	$T \rightarrow T\ Prod\ F$
7	$E \rightarrow D$	$E \rightarrow T$
8	$E \rightarrow D\ Sum\ D$	$E \rightarrow E\ Sum\ T$

Figure 6: Examples of $P_\perp$ and learned $P_\top$

$ch_\perp$, where $B_i \rightarrow \beta_\perp \overset{B_j \rightarrow \beta_\perp \subset}{\dashv} B_i \rightarrow B_j$ holds for all $B_j \prec B_i$. Thus, keeping (or recovering) the ordered $ch_\top$ in any grammar G derived from $\top$, all the other grammar rules are determinate generalizable. □

We now introduce Algorithm 1, which given the representative set E_R and the generalization set E_{gen}, builds the $\perp$ and $\top$ grammars. First, an assumption in our learning model is that the rules corresponding to the grammar preterminals (P_Σ) are given. Thus, for a given representative set E_R, we can build the grammar $\perp$ in the following way: for each observable $\langle w, A\rangle \in E_R$ the category A gives the name of the left-hand side nonterminal of the grammar rule, while the right-hand side is constructed using a bottom-up active chart parser (Kay, 1973) (line 1 in Algorithm 1). For our mathematical expressions example, given E_R in Figure 2 and P_Σ in Figure 1, the rules of the bottom grammar $P_\perp$ are given in Figure 6.

Algorithm 1 `Top` (E_R, E_{gen})

1: $P_\perp \leftarrow Bottom(E_R)$
2: $P_\top \leftarrow$ `Chains_Recovery` $(P_\perp, E_R, E_{gen})$ $\{P_\top$ is determinate generalizable $\}$
3: **while** $\exists r \in P_\top\ s.t.\ r \overset{1\subset}{\dashv} r_g$ **do**
4: $\quad r \leftarrow r_g$;
5: **end while**
6: **return** $P_\top$

In order to build the $\top$ element (lines 2-5 in Algorithm 1), we first need to apply the `Chain_recovery` algorithm to the lattice $\perp$ grammar (line 2 in Algorithm 1). `Chains_Recovery` first detects all $ch = ch_\perp$ which contain rules with identical right-hand side (line 5-6). Then, all $ch_\perp$ rules are transformed in $ch_\top$ by generalizing them through the minimal chain rule (lines 10-17). The generalization step $r \overset{r_m \subset}{\dashv} r_g$ guarantees the semantic increase $\mathbb{S}(r_g) \supset \mathbb{S}(r)$ for all the rules r which are generalized through r_m, thus being the inverse of the rule specialization step in the grammar lattice. The rules r are either chain rules or rules having the same left-hand side as the chain rules. The returned set $P_\perp$ contains all unary branching rules ($ch_\top$) of the $\top$ grammar. The efficiency of `Chain_Recovery` algorithm is $O(|E_R| * |\beta| * |E_{gen}|)$. Therefore, in Algorithm 1 the set $P_\top$ initially contains determinate generalizable rules and the `while loop` (lines 3-5, Alg 1) can determinately generalize all the grammar rules. In Figure 6, the grammar $P_\top$ is learned by Algorithm 1, based only on the examples in Figure 2 and P_Σ in Figure 1.

Algorithm 2 `Chains_Recovery`

1: **Input:** $P_\perp, E_R, E_{gen}$
2: **Output:** $P_\perp$ which contains all $ch_\top$
3: **while** $E_R \neq \emptyset$ **do**
4: $\quad \langle w, c\rangle \leftarrow first(E_R)$
5: $\quad ch \leftarrow \{r \in P_\perp | r \overset{*\perp}{\Rightarrow} w\}$ $\{ch = ch_\perp\}$
6: $\quad lch \leftarrow \{lhs(r) | r \in ch\}$
7: $\quad$ **for each** $c \in lch$ **do**
8: $\quad\quad E_R \leftarrow E_R - \{\langle w, c\rangle\}$
9: $\quad$ **end for**
10: $\quad$ **while** $|ch| > 1$ **do**
11: $\quad\quad r_m \leftarrow$ `MinRule(ch)`
12: $\quad\quad ch \leftarrow ch - \{r_m\}$
13: $\quad\quad lch \leftarrow lch - \{lhs(r_m)\}$
14: $\quad\quad$ **for each** $r \in P_\perp \wedge lhs(r) \in lch\ s.t.\ r \overset{r_m \subset}{\dashv} r_g$ **do**
15: $\quad\quad\quad r \leftarrow r_g$
16: $\quad\quad$ **end for**
17: $\quad$ **end while**
18: **end while**
19: **return** $P_\perp$

Algorithm 3 `MinRule`

1: **Input:** the chain ch
2: **for each** $r_m \in ch$ **do**
3: $\quad find \leftarrow true$
4: $\quad$ **for each** $r \in ch - \{r_m\}$ **do**
5: $\quad\quad$ **if** $r_m \overset{r}{\dashv} r_{mg} \wedge \mathbb{S}(r_m) \subset \mathbb{S}(r_{mg})$ **then**
6: $\quad\quad\quad find \leftarrow false$
7: $\quad\quad$ **end if**
8: $\quad$ **end for**
9: $\quad$ **if** $find == true$ **then**
10: $\quad\quad$ **return** r_m
11: $\quad$ **end if**
12: **end for**

Theorem 2 (Learnability Theorem)**.** *If E_R is*

the representative set of a WFG G conformal w.r.t a generalization set $E_{gen} \supseteq E_R$, then Top(E_R, E_{gen}) algorithm computes the lattice $\top$ element such that $\mathbb{S}(\top) = E_{gen}$.

Proof. Since G is normalized, none of its rule can be generalized with increase in semantics. Starting with the $\bot$ element, after `Chains_Recovery` all rules that can be generalized with semantic increase through the rule generalization step, are determinate generalizable. This means that the grammar generalization sequence $\bot, G_1, \ldots, G_n, \top$, ensures the semantic increase of $\mathbb{S}(G_i)$ so that the generalization process ends at the semantic limit $\mathbb{S}(\top) = E_{gen}$. □

For WFGs which have rules that can be either left or right recursive, the top element is unique only if we impose a direction of generalization in the rule's right-hand side (e.g., left to right). Another way to guarantee uniqueness of the top element is to add constraints at the grammar rules. In our example, if we augment de grammar nonterminals with expression values (semantic interpretation) and we add constraints at the grammar rules we have $E(v) \rightarrow E(v_1)\ Sum(op)\ T(v_2)$: $\{v \leftarrow v_1\ op\ v_2\}$. With the generalization example $\langle 5-3-1, E(1)\rangle \in E_{gen}$ we can generalize the rule $E \rightarrow T\ Sum\ T$ only to $E \rightarrow E\ Sum\ T$ and not to $E \rightarrow T\ Sum\ E$ because $1 = (5-3)-1$ and $1 \neq 5-(3-1)$. For our Lexicalized Well-Founded Grammars this problem is solved by associating strings with their syntactic-semantic representations and by having semantic compositional constraints at the grammar rule level.

3 Conclusions

In this paper, we discussed the learnability of Lexicalized Well-Founded Grammars. We introduced the class of well-founded grammars and presented the theoretical underpinnings for learning these grammars from a representative set of positive examples. We proved that under several assumptions the search space for learning these grammars is a complete grammar lattice. We presented a general algorithm which builds the top and the bottom elements of the complete grammar lattice and gave a learnability theorem. The theoretical results obtained in this paper hold for the LWFG formalism, which is suitable for deep linguistic processing.

References

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4).

Alexander Clark. 2006. PAC-learning unambiguous NTS languages. In *Proceedings of the 8th International Colloquium on Grammatical Inference (ICGI 2006)*, pages 59–71.

Marc Denecker, Maurice Bruynooghe, and Victor W. Marek. 2001. Logic programming revisited: Logic programs as inductive definitions. *ACM Transactions on Computational Logic*, 2(4):623–654.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of the ACL '02*, pages 335–342.

Aravind Joshi and Yves Schabes. 1997. Tree-Adjoining Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 2, pages 69–124. Springer, Berlin,New York.

Martin Kay. 1973. The MIND system. In Randall Rustin, editor, *Natural Language Processing*, pages 155–188. Algorithmics Press, New York.

Smaranda Muresan and Owen Rambow. 2007. Grammar approximation by representative sublanguage: A new model for language learning. In *Proceedings of ACL'07*.

Smaranda Muresan. 2006. *Learning Constraint-based Grammars from Representative Examples: Theory and Applications*. Ph.D. thesis, Columbia University.

Smaranda Muresan. 2011. Learning for deep language understanding. In *Proceedings of IJCAI-11*.

Fernando C. Pereira and Stuart M. Shieber. 1984. The semantics of grammar formalisms seen as computer languages. In *Proceeding of the ACL 1984*.

Fernando C. Pereira and David H.D Warren. 1980. Definite Clause Grammars for language analysis. *Artificial Intelligence*, 13:231–278.

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, Illinois.

Libin Shen. 2006. *Statistical LTAG Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA. AAI3225543.

Alfred Tarski. 1955. Lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309.

Maarten H. van Emden and Robert A. Kowalski. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742.

Shuly Wintner. 1999. Compositional semantics for linguistic formalisms. In *Proceedings of the ACL'99*.