Lexicalized Well-Founded Grammars: Learnability and Merging

Smaranda Muresan SMARA@CS.COLUMBIA.EDU
Department of Computer Science
Columbia University
500 West 120th St, New York, NY

Tudor Muresan TMURESAN@CS.UTCLUJ.RO
Department of Computer Science
Technical Univ. of Cluj-Napoca
Cluj-Napoca, Romania

Judith Klavans KLAVANS@CS.COLUMBIA.EDU
Department of Computer Science
Columbia University
500 West 120th St, New York, NY

Abstract

This paper presents the theoretical foundation of a new type of constraint-based grammars, *Lexicalized Well-Founded Grammars*, which are adequate for modeling human language and are learnable. These features make the grammars suitable for developing robust and scalable natural language understanding systems. Our grammars capture both syntax and semantics and have two types of constraints at the rule level: one for semantic composition and one for ontology-based semantic interpretation. We prove that these grammars can always be learned from a small set of semantically annotated, ordered representative examples, using a relational learning algorithm. We introduce a new semantic representation for natural language, which is suitable for an ontology-based interpretation and allows us to learn the compositional constraints together with the grammar rules. Besides the learnability results, we give a principle for grammar merging. The experiments presented in this paper show promising results for the adequacy of these grammars in learning natural language. Relatively simple linguistic knowledge is needed to build the small set of semantically annotated examples required for the grammar induction.

Keywords Constraint-based grammar induction, inductive logic programming, ontology-based semantic representation, natural language understanding.

1. Introduction

Computer-understanding of human language has been for a long time one of the main challenges of artificial intelligence. Research in the area of natural language understanding has usually followed two mutually exclusive paths: deep, rule-based, linguistically motivated approaches and shallower, machine learning approaches. The former has focused on developing broad coverage grammars able to capture both syntax and semantics, complex lexicons, and complex modules for deep semantic interpretation. However, these lexicons and grammars have been hand-crafted, very hard to scale-up, and involved long time effort by large teams of linguists and computational linguists (e.g., LKB

(Copestake, 1999), ALE (Carpenter & Penn, 1999), XTAG (Paroubek, Schabes, & Joshi, 1992)). Machine learning approaches, on the other hand, have been applied to restricted domains (e.g., air travel domain (Miller, Bobrow, Ingria, & Schwartz, 1994; Macherey, Och, & Ney, 2001)) or tasks (e.g., information extraction (Cardie, 1997; Jones, Ghani, Mitchell, & Riloff, 2003), and acquisition of semantic lexicons (Siskind, 2000; Thompson & Mooney, 2003)).

In order to bridge the gap between deep language processing and machine learning, current research focuses on building large, richer treebanks, annotated with both syntactic and semantic information. On the one hand, there are PropBank (Kingsbury, Palmer, & Marcus, 2002), and FrameNet (Baker, Fillmore, & Lowe, 1998). These resources have led to advances in applying machine learning methods for the task of semantic role labeling, which can be regarded as a first step towards deeper language understanding (Gildea & Jurafsky, 2002; Chen & Rambow, 2003; Carreras & Marquez, 2004). On the other hand, there are resources such as Redwoods (Oepen, Flickinger, Toutanova, & Manning, 2002), whose goal is to build treebanks for deep linguistics frameworks like Head-Driven Phrase Structure Grammars (Pollard & Sag, 1994). The HPSG formalism has been widely used in deep, linguistically motivated language understanding systems. The Redwoods treebank has been used for stochastic modeling of a hand-built HPSG grammar, English Resource Grammar (Flickinger, 2000), to reduce its ambiguity. However, when applied to real corpora, the coverage of these hand-built grammars is low.

Even though building such large, complex annotated corpora implies arguably less work than building the grammar, it is not a simple task. Moreover, the complex representations of existing grammar formalisms used in language understanding (e.g., HPSG, LFG) pose challenges to learning methods. Thus, research on grammar learning has focused mainly on syntax, using both supervised and unsupervised methods (Klein & Manning, 2001; Collins, 1999; Osborne, 1999). Few efforts have been made for inducing grammars that capture both form and meaning, such as Attribute Grammars (Starkie, 2002), and Categorial Grammars (Retore & Bonato, 2001; Dudau-Sofronie, Tellier, & Tommasi, 2001). However, the semantics of Categorial Grammars adheres to the truth-conditional theory of semantics based on $\lambda - calculus$, which has been recently argued to be unsuitable both from linguistic and computational considerations (Dalrymple, 1999; Copestake, Lascarides, & Flickinger, 2001).

In this paper we discuss a type of constraint-based grammars, *Lexicalized Well-Founded Grammars*, and present several new theoretical properties for their learnability and a principle for their merging. These grammars facilitate a new approach to natural language understanding, which integrates deep semantic analysis and relational learning.

Lexicalized Well-Founded Grammars, introduced in our previous work (Muresan, Muresan, & Klavans, 2004; Muresan, 2004), are constraint-based grammars, which associate a syntactic-semantic representation to each string, and have two types of constraints at the rule level — one for semantic composition and one for ontology-based interpretation. The presence of the ontology at the grammar rule level provides access to meaning during parsing and during grammar rule learning. This design follows theories of human language acquisition, which argue that access to meaning is needed for language learning (Pinker, 1989; Culicover & Nowak, 2003). Lexicalized Well-Founded Grammars allow a deep semantic analysis of natural language.

Relational rule learning was discouraged by Cohen's negative results regarding the PAC-learnability of recursive logic programs (Cohen, 1995). Cohen showed that even two-clause linear recursive logic programs are not PAC-learnable from random examples, in the absence of an oracle. In (Muresan, Muresan, & Potolea, 2002) we showed that introducing an order among examples makes the

learning process tractable. Thus, complex logic programs can be learned bottom-up from ordered representative examples, based only on the declaration of predicate calling graph and the predicate arguments data flow. Using this relational learning method, in (Muresan et al., 2004) we presented a framework for inducing Lexicalized Well-Founded Grammars from a small set of ordered representative examples. These examples are the simplest pairs of strings and their syntactic-semantic representations, which are generated by the grammar. The ordering of examples allows for an efficient bottom-up relational learning of the grammar rules and their compositional semantic constraints, simulating child language learning from simple constructions to complex ones (Pinker, 1989). The small size of the representative example set is an important practical advantage, since semantic annotations are not readily available and are hard to build for a variety of domains.

In this paper, we define a new type of Lexicalized Well-Founded Grammars, called *Normalized Lexicalized Well-Founded Grammars* that are conform to a sublanguage. We report two major theoretical results. First, we show that for these grammars the sublanguage mediates the reciprocal generation of representative examples and grammar rules. Thus, on the one hand, given the grammar and the sublanguage, the representative examples are generated. On the other hand, we prove a learnability theorem, which states that a Normalized Lexicalized Well-Founded Grammar can always be learned from the set of representative examples and a sublanguage that is conform to the grammar. The sublanguage is used to reduce the grammar semantics, which is used as performance criterion in the Inverse Entailment learning method (Muggleton, 1995; Muresan et al., 2002). Second, we give a grammar merging principle that shows how several grammars can be merged in a sound way, by the union of their representative examples, their sublanguages and the subsequent use of the grammar learning algorithm. We also show that grammar merging does not consist merely in the union of their production rules. This theoretical result addresses one of the major concerns regarding grammar development and engineering: how can grammar modularity be defined formally, so that different fragments of grammars can be combined together in a sound way (Wintner, 2002).

The experiments presented in this paper show promising results for these grammars' adequacy in learning natural language. At the same time, relatively simple linguistic knowledge is required to build the small set of semantically annotated examples.

The rest of the paper is organized as follows. In Section 2, we define formally the Well-Founded Grammars (WFGs). In Section 3, we present the Lexicalized Well-Founded Grammars that augment the WFGs with semantics. Here, we first describe a new semantic representation — semantic molecule – and our approach for semantic composition and semantic interpretation modeled as constraints at the grammar rule level. Then, we define this new type of constraint-based grammars, including the notion of derivation in LWFGs, grammar semantics and reversible, robust parsing. In Section 4, we introduce the notion of representative examples associated with a grammar and a sublanguage, and we present an efficient algorithm for their generation. In Section 5, we discuss the learnability of Normalized Lexicalized Well-Founded Grammars, including the impact of their properties for learnability, several assumptions of these grammars, the relational learning algorithm for grammar induction and the learnability theorem. We also present an iterative algorithm for grammar revision. In Section 6, we present the grammar merging principle, and in Section 7 we describe our results and the discussion of linguistic relevance. We conclude with a summary of the paper, and future work.

2. Well-Founded Grammars

Well-Founded Grammars were introduced by Muresan, Muresan and Klavans (2004). These grammars extend Context-Free Grammars by introducing a partial ordering relation among the nonterminals. This allows the ordering of the strings derived from the grammar and thus an ordering of the grammar rules. This property is meant to facilitate the bottom-up induction of these grammars.

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

(i) Σ is a finite set of terminal symbols.

(ii) N_G is a finite set of nonterminal symbols, $N_G \cap \Sigma = \emptyset$.[1]

(iii) R_G is a partial ordering relation, $\succeq$, among the nonterminals.

(iv) P_G is a set of production rules, where each rule is an element of $N_G \times \{N_G \cup \Sigma\}^+$. The rule $(A, (B_1, \ldots, B_n))$ is written $A \rightarrow B_1, \ldots, B_n$. Sometimes, for brevity, we denote a rule by $A \rightarrow \beta$, where $\beta = B_1, \ldots, B_n$. The set of production rules has the following characteristics:

- There are three types of rules: ordered non-recursive rules, ordered recursive rules, and non-ordered rules. We call a rule $(A \rightarrow B_1, \ldots, B_n) \in P_G$, an *ordered rule*, if $\forall B_i$, we have $A \succeq B_i$.
- Every nonterminal symbol is a left-hand side in at least one ordered non-recursive rule.
- The empty string cannot be derived from any nonterminal symbol.

(v) $S \in N_G$ is the start nonterminal symbol and $\forall A \in N_G, S \succeq A$ [2].

Definition 2. Given a Well-Founded Grammar $G = \langle \Sigma, N_G, R_G, P_G, S \rangle$, the *ground derivation* [3], $\stackrel{*}{\Rightarrow}$, is defined as: $\frac{A \rightarrow w}{A \stackrel{*}{\Rightarrow} w}$ (if A is a grammar preterminal, i.e., $w \in \Sigma$), and $\frac{B_i \stackrel{*}{\Rightarrow} w_i,\ i=1,\ldots,n, \quad A \rightarrow B_1,\ldots,B_n}{A \stackrel{*}{\Rightarrow} w}$, where $w = w_1 \cdots w_n$.

The *language* of a grammar G is $L(G) = \{w | w \in \Sigma^+, S \stackrel{*}{\Rightarrow} w\}$. The set of all strings generated by a grammar G is $L_s(G) = \{w | w \in \Sigma^+, \exists A \in N_G, A \stackrel{*}{\Rightarrow} w\}$. We have that $L(G) \subseteq L_s(G)$. Extending the notation, given a grammar G, the set of strings generated by *a nonterminal* A of the grammar G is $L_s(A) = \{w | w \in \Sigma^+, A \in N_G, A \stackrel{*}{\Rightarrow} w\}$, and the set of strings generated by *a rule* $A \rightarrow \beta$ of the grammar G is $L_s(A \rightarrow \beta) = \{w | w \in \Sigma^+, (A \rightarrow \beta) \stackrel{*}{\Rightarrow} w\}$[4] .

Every Context-Free Grammar, $G = \langle \Sigma, N_G, P_G, S \rangle$ can be efficiently tested to see whether it is a Well-Founded Grammar, by Algorithm 1. This algorithm assigns one and only one level l to every nonterminal A, $A \in N_G^l$, and returns the set of partial ordered pairs of nonterminals, R_G. We

1. We use lower-case letters for terminal symbols and upper-case letters for nonterminal symbols.
2. We used the same notation for the reflexive, transitive closure of $\succeq$.
3. The ground derivation ("reduction" in (Wintner, 1999)) can be viewed as the bottom-up counterpart of the usual derivation.
4. We use the notation $(A \rightarrow \beta) \stackrel{*}{\Rightarrow} w$ to denote the derivation $A \stackrel{*}{\Rightarrow} w$ obtained using the rule $A \rightarrow \beta$ in the last derivation step.

denote by N_G^l the set of nonterminals assigned to the level l, with $l \geq 1$. The set of terminals are assigned to level 0, denoted by N_G^0 to keep an analogous notation. The efficiency of the algorithm is O($|P_G|^2 * |\beta|$) [5].

Algorithm 1: Well_Founded_Grammar($G = \langle \Sigma, N_G, P_G, S \rangle$)

$R_G \leftarrow \emptyset$, $N_G^0 \leftarrow \Sigma$, $P \leftarrow P_G$, $V \leftarrow \emptyset$, $l \leftarrow 0$
while $P \neq \emptyset$ *and* $N_G^l \neq \emptyset$ **do**
  $V \leftarrow V \cup N_G^l$, $l \leftarrow l + 1$, $N_G^l \leftarrow \emptyset$
  foreach $(A \rightarrow \beta) \in P$ *and* $\beta \in V^+$ **do**
    $P \leftarrow P - \{A \rightarrow \beta\}$
    if $A \notin V$ **then**
      $N_G^l \leftarrow N_G^l \cup \{A\}$
      foreach $(B \in N_G$ *and* $B \in \beta)$ **do**
        $R_G \leftarrow R_G \cup \{A \succeq B\}$
    else
      foreach $(B \in N_G$ *and* $B \in \beta)$ *and not*$(A \succeq B$ *or* $B \succeq A)$ **do**
        if $A \in N_G^i$ *and* $B \in N_G^j$ *and* $i \geq j$ **then**
          $R_G \leftarrow R_G \cup \{A \succeq B\}$
        else
          $R_G \leftarrow R_G \cup \{B \succeq A\}$
if $P = \emptyset$ **then** return R_G **else** return $\emptyset$

A nonterminal is assigned to a level, when it appears on the left-hand side of an ordered non-recursive rule (Figure 1). The algorithm guarantees that if $A \in N_G^i$, $B \in N_G^j$, $i \geq j$, and if there exists a direct relation between A and B, then this relation is $A \succeq B$. This property states that if a direct relation exists, the nonterminals on the "upper" levels are bigger than the nonterminals on the "lower" levels. For the nonterminals on the same level i, $A, B, C \in N_G^i$, the partial ordering relation, if it exists, depends on the order of processing the grammar rules.

Lemma 1. *A Context-Free Grammar $G = \langle \Sigma, N_G, P_G, S \rangle$ is a Well-Founded Grammar $G = \langle \Sigma, N_G, R_G, P_G, S \rangle$ iff $R_G \neq \emptyset$ is returned by Algorithm 1.*

Proof. The proof is immediate.

Example. Figure 1 gives an example of a small grammar for noun phrases and the iterative steps of Algorithm 1. As can be seen, in this grammar, $A1 \rightarrow Adj$, $N1 \rightarrow Noun$, $N2 \rightarrow Det \;\; N1$ are examples of ordered non-recursive rules; $N1 \rightarrow A1 \;\; N1$ is an example of an ordered recursive rule, while $N2 \rightarrow N2 \;\; Rc1$ is a non-ordered rule, since $Rc1$ is a bigger nonterminal than $N2$, i.e., $Rc1 \succeq N2$ (see Figure 1(b)). We use Rc to denote relative clauses.

By introducing a partial ordering relation among nonterminals, the Well-Founded Grammars can provide a partial ordering among the strings derived by these grammars (see Section 4).

5. We use the same notation $|\cdot|$ for the number of set elements and for the string length.

(a)

$A1 \rightarrow Adj$
$N1 \rightarrow Noun$
$N1 \rightarrow A1\ \ N1$
$N2 \rightarrow Det\ \ N1$
$N2 \rightarrow N2\ \ Rc1$
$V1 \rightarrow Tv$
$Rc1 \rightarrow Rpro\ V1\ N2$

(b)

P_G (grammar rules)	l (level)	N_G^l (nonterminal sets)	R_G (partial ordering relation)
	1	$\{Adj, Noun, Tv, Det, Rpro\}$	
$A1 \rightarrow Adj$ $N1 \rightarrow Noun$ $V1 \rightarrow Tv$	2	$\{A1, N1, V1\}$	$A1 \succeq Adj$ $N1 \succeq Noun$ $V1 \succeq Tv$
$N1 \rightarrow A1\ N1$ $N2 \rightarrow Det\ N1$	3	$\{N2\}$	$N1 \succeq A1, N1 \succeq N1$ $N2 \succeq Det, N2 \succeq N1$
$Rc1 \rightarrow Rpro\ V1\ N2$	4	$\{Rc1\}$	$Rc1 \succeq Rpro,$ $Rc1 \succeq V1, Rc1 \succeq N2$
$N2 \rightarrow N2\ Rc1$			$N2 \succeq N2$

Figure 1: (a) Grammar, G; (b) Iterative steps for the Well_Founded_Grammar(G) algorithm

3. Augmenting Well-Founded Grammars with Semantics

Augmenting a grammar with semantics requires an adequate semantic representation, and an adequate grammar formalism that allows us to associates structures to nonterminals and to add constraints at the grammar rule level.

In the remainder of this section we present our new syntactic-semantic representation, our approach for encoding the semantic composition and the semantic interpretation as constraints at the grammar rule level, and our new type of constraint-based grammars, *Lexicalized Well-Founded Grammars*.

3.1 Semantic Molecule

The underlying design criteria for our representation are the need to: 1) explicitly encode the information for semantic composition (i.e., how the meaning of the whole is derived from the meaning of its parts), 2) capture the semantics of a natural language expression, so that it is suitable for an ontology-based interpretation, 3) use simple representation devices so that they can be integrated in a relational learning algorithm, and 4) link the semantic construction to other grammatical aspects, most notably syntax. We introduce a new representation, called *semantic molecule*, which satisfies the above considerations.

Definition 3. A *semantic molecule* associated with a natural language string w, is a syntactic-semantic representation, $w' = h \bowtie b$, where:

- h ($head$) encodes syntactic/compositional information, acting as valence for molecule composition.
- b ($body$) is the actual semantic representation of the string w.

Figure 2 shows examples of semantic molecules for an adjective (I-1), a noun (I-2) and a noun phrase (II). The representations associated with the lexical items $w \in \Sigma$ are called *elementary semantic molecules* (I), while the representations built by the combination of others are called *derived semantic molecules* (II). We will describe the composition operation, $\circ$, which combines several semantic molecules to form a derived semantic molecule in the next section.

I. Elementary Semantic Molecules

$\mathbf{I-1}$ $(tall/adj)' = h_1 \bowtie b_1$

$$= \begin{bmatrix} cat & adj \\ head & X_1 \\ mod & X_2 \end{bmatrix} \bowtie [X_1.isa = tall, X_2.Y = X_1]$$

$h_1.cat = adj$
$h_1.head = X_1$
$h_1.mod = X_2$
$\mathcal{A}_{h_1} = \{cat, head, mod\}$
$var(h_1) = \{X_1, X_2\}$
$var(b_1) = \{X_1, X_2, Y\}$

$\mathbf{I-2}$ $(man/n)' = h_2 \bowtie b_2$

$$= \begin{bmatrix} cat & n \\ nr & sg \\ head & X_3 \end{bmatrix} \bowtie [X_3.isa = man]$$

$h_2.cat = n$
$h_2.nr = sg$
$h_2.head = X_3$
$\mathcal{A}_{h_2} = \{cat, nr, head\}$
$var(h_2) = \{X_3\}, var(b_2) = \{X_3\}$

II. Derived Semantic Molecule

$(tall\ \ man)' = h \bowtie b = (tall)' \circ (man)'$

$$= \begin{bmatrix} cat & n \\ nr & sg \\ head & X \end{bmatrix} \bowtie [X_1.isa = tall, X.Y = X_1, X.isa = man]$$

$h.cat = n$
$h.nr = sg$
$h.head = X$
$\mathcal{A}_h = \{cat, nr, head\}$
$var(h) = \{X\},$
$var(b) = \{X_1, X, Y\}$

III. Constraint Grammar Rule Associated with the Derived Semantic Molecule

$N(w, h \bowtie b) \rightarrow Adj(w_1, h_1 \bowtie b_1),\ N(w_2, h_2 \bowtie b_2)\ :$
$w = w_1 w_2,\ b = [b_1, b_2]\nu,\ \Phi_{comp}(h, h_1, h_2),\ \Phi_{onto}(b)$

$$\Phi_{comp}(h, h_1, h_2) = (h \cup h_1 \cup h_2)\mu\nu = \left\{ \begin{array}{l} h.cat = n, \\ h.head = h_1.mod, \\ h.head = h_2.head, \\ h.nr = h_2.nr, \\ h_1.cat = adj, \\ h_2.cat = n \end{array} \right\}$$

$\nu = \{X_2/X, X_3/X\}$

$\mu = \{h.nr = sg/h.nr = X_4, h_2.nr = sg/h_2.nr = X_4\}$

Figure 2: Augmenting grammars with semantics. Examples of two elementary semantic molecules for an adjective (I-1: $(tall)'$) and a noun (I-2: $(man)'$), and a derived semantic molecule obtained by combining them (II: $(tall\ man)'$). In (III) is given the constraint grammar rule used to derive the string $w = tall\ man$ together with its semantic representation $w' = (tall\ man)'$. The compositional semantic constraint, Φ_{comp}, together with the variable and contextual constant substitutions, ν, and μ, are also shown. The Xs and Ys denote logical variables, while lower-case letters denote constants. In the case of semantic molecules, $w' = h \bowtie b$, the lower-case letters indicate they are grounded (i.e., they are directly associated with a ground string, w), even if they contain logical variables.

The head, h, of a semantic molecule is represented as a one level feature structure (i.e., feature values are atomic). In Figure 2 the heads are shown as attribute-value matrices (AVMs). Let $\mathcal{A}_h$ be the set of attributes of a molecule head, h. Each molecule has at least two attributes encoding the syntactic category of the associated string, cat, and the head of the string, $head$. For adjectives, for example, besides these two attributes, there is an attribute, mod, which specifies the index of the modified noun (I-1). This information is necessary for combining an adjective and a noun to obtain a noun phrase (e.g., "tall man"). For nouns, we can have other syntactic information (e.g., nr) that will be used for agreement (e.g., number agreement between the subject and the main verb of a sentence). All these sets of attributes are finite and are known a priori for each syntactic category. The elements of h are denoted as $h.a = val$, where $a \in \mathcal{A}_h$ and val is either a constant or a logical variable (see I-1 and I-2). For example $h.cat = adj$, and $h.cat = n$ denote the syntactic categories of the semantic molecules for "tall" (adjective) and "man" (noun), respectively. The set of logical variables of the head, h, is denoted by $var(h)$.

The body, b, of a semantic molecule is a flat representation (i.e., no embedding of predicates is allowed, as in Minimal Recursion Semantics (Copestake, Flickinger, & Sag, 1999)), called Canonical Logical Form (CLF). It is built using a set of atomic predicates (APs) based on the concept of attribute-value pair:

(1)
$$\begin{aligned} \langle \text{CLF} \rangle &\longrightarrow \langle \text{AP} \rangle \\ &\quad\ \mid \langle \text{CLF} \rangle \, lop \, \langle \text{CLF} \rangle \\ \langle \text{AP} \rangle &\longrightarrow \langle \text{concept} \rangle \ . \ \langle \text{attr} \rangle = \langle \text{concept} \rangle \end{aligned}$$

The *lop* is a logical operator, while *concept* is a frame in the ontology and *attr* is a slot of the frame, encoding either a property or a relation. As seen in Figure 2 (I-1 and I-2), our semantic representation is influenced by the ontology-based approach to semantic interpretation. Thus, CLF can be seen as an Ontology Query Language. For example, in our framework, the meaning of a noun is the corresponding basic concept in the ontology ($X_3.isa = man$). The meaning of an adjective is the concept corresponding to a value of a property (slot) of another concept denoted by a noun ($X_1.isa = tall, X_2.Y = X_1$). The set of logical variables of the body, b, are denoted by $var(b)$. For adjective, $var(b) = \{X_1, X_2, Y\}$, where X_2 will be bound to the head of the modified noun after the composition operation (e.g., X_2 will be the same as the head X_3 of the noun "man" after the composition that derives "tall man"), while the variable Y will be instantiated after the semantic interpretation on the ontology (e.g., for the noun phrase "tall man", $Y = height$). The semantic composition and the semantic interpretation are discussed in the next section.

3.2 Semantic Composition and Semantic Interpretation as Grammar Constraints

A requirement for computational semantic frameworks, besides linguistic adequacy and computational tractability, is grammar compatibility (Copestake et al., 1999). This refers to the ability of the semantic construction to be connected to other grammatical aspects, mainly syntax. Constraint-based grammar frameworks have been widely use to capture both aspects of syntax and semantics. In particular, the Definite Clause Grammar formalism (Pereira & Warren, 1980) extends the Context-Free Grammars in three important ways: 1) it allows for context-dependency in a grammar; 2) it allows us to build arbitrary structures during parsing, in a way that is not constrained by the recursive structure of the grammar (such structures can provide the *meaning* of the string); and 3) it allows extra conditions to be included in the grammar rules, that can be seen as constraints for

parsing. The first and second mechanism are provided in the DCG formalism by augmenting the nonterminals with extra arguments. Thus, DCG is a suitable formalism for our purpose, since it allows us to augment the nonterminals with pairs of strings and their *semantic molecules*, and to introduce two types of constraints at the grammar rule level — one for semantic composition (defines how the meaning of a natural language expression is composed from the meaning of its parts) and one for ontology-based semantic interpretation (validates the semantic constructions at the rule level).

Definition 4. A *generalized syntagma*, $\sigma = (w, w')$, is a pair of a natural language string and its semantic molecule, and represents any unit that can be derived from a grammar. It refers to words, phrases, clauses and sentences.

Thus, the nonterminals of a constraint grammar rule are augmented with generalized syntagmas, the grammar rules having the following form: [6]

$$A(w, \overbrace{h \bowtie b}^{w'}) \rightarrow B_1(w_1, \overbrace{h_1 \bowtie b_1}^{w'_1}), ..., B_n(w_n, \overbrace{h_n \bowtie b_n}^{w'_n}): \\ w = w_1 \cdots w_n,\ \underbrace{b = [b_1, \ldots, b_n]\nu,\ \Phi_{comp}(h, h_1, ..., h_n)}_{w' = w'_1 \circ \cdots \circ w'_n},\ \Phi_{onto}(b)$$

where:

- $A, B_1, \ldots, B_n$ - grammar nonterminals, which represent syntactic categories ($A = h.cat,\ B_i = h_i.cat$).
- $w, w_1, \ldots, w_n$ - natural language strings.
- $w' = h \bowtie b, w'_1 = h_1 \bowtie b_1, \ldots, w'_n = h_n \bowtie b_n$ - semantic molecules corresponding to the natural language strings $w, w_1, ...,$ and w_n, respectively.
- : - delimiter for constraints.
- $\circ$ - composition of semantic molecules: $b = [b_1, \ldots, b_n]\nu, \Phi_{comp}(h, h_1, ..., h_n)$, where $\Phi_{comp}(h, h_1, \ldots, h_n) = [(h \cup \bigcup_{1 \leq i \leq n} h_i)\mu]\nu$ (see Section 3.2.1).
- ν, μ - variable and contextual constant substitutions (see Section 3.2.1).
- $\Phi_{onto}(b)$ - ontology-based semantic interpretation constraint applied only to the body of the semantic molecule corresponding to the left-hand side nonterminal (see Section 3.2.2).

As can be seen, both the strings and their semantic molecules are attached to nonterminals and each grammar rule is enhanced with the following constraints: the string composition as concatenation of strings ($w = w_1 \cdots w_n$), the semantic composition of molecules given by "$\circ$" ($w' = w'_1 \circ \cdots \circ w'_n$ = [$b = [b_1, \ldots, b_n]\nu, \Phi_{comp}(h, h_1, ..., h_n)$]) and the ontology-based semantic interpretation constraint, $\Phi_{onto}(b)$. An example of a grammar rule for noun phrases that contain nouns modified by adjectives is given in Figure 2.

6. For the clarity of the presentation we keep the below notation and not the DCG notation. In our implementation, both the concatenation of strings, $w = w_1 \cdots w_n$, and the concatenation of their semantic representations, $b = b_1 \cdots b_n$ are implemented as Prolog difference lists (see examples of grammar rules given in Appendix A and B).

3.2.1 SEMANTIC COMPOSITION CONSTRAINTS

The semantic composition, $\circ$, defines how the representation of a natural language expression (corresponding to the left-hand side nonterminal, A) is composed from the representations of its parts:

$$
\begin{aligned}
w' = h \bowtie b = (w_1 \cdots w_n)' &= w'_1 \circ \cdots \circ w'_n \\
&= (h_1 \bowtie b_1) \circ \cdots \circ (h_n \bowtie b_n) \\
&= h_1 \circ \cdots \circ h_n \bowtie b_1, \ldots, b_n
\end{aligned} \tag{2}
$$

The composition affects only the molecule heads and it is realized by a set of constraints denoted by $\Phi_{comp}(h, h_1, ..., h_n)$. The body parts are connected through conjunction ($b = b_1, \ldots, b_n$). From now on, when referring to semantic composition we mean both Φ_{comp} and the body conjunction, while when referring to *compositional semantic constraints* we mean just Φ_{comp}.

This set of constraints, $\Phi_{comp}(h, h_1, ..., h_n)$, is encoded as a system of equations, (3a), (3b) and will be learned together with the grammar rule during the induction process.

$$
\left\{ \begin{array}{l} h.a = constant \\ h.a = h_i.a_i \end{array} \right\} \quad \text{where} \begin{array}{c} 1 \leq i \leq n \\ a \in \mathcal{A}_h, a_i \in \mathcal{A}_{h_i} \end{array} \tag{3a}
$$

$$
\left\{ \begin{array}{l} h_i.a_i = constant \\ h_i.a_i = h_j.a_j \end{array} \right\} \quad \text{where} \begin{array}{c} 1 \leq i, j \leq n, i \neq j \\ a_i \in \mathcal{A}_{h_i}, a_j \in \mathcal{A}_{h_j} \end{array} \tag{3b}
$$

In Figure 2 (II), we give an example of semantic composition for the noun phrase "tall man", obtained by composing the semantic molecules of the adjective "tall" (I-1) and the noun "man" (I-2). The grammar rule associated with this derived molecule, together with the compositional semantic constraints, $\Phi_{comp}(h, h_1, h_2)$, are also given (III). As a consequence of variable bindings due to head composition, some variables from the bodies of the semantic molecules are bound as well (e.g., the variables X_2 and X_3 are bound). It can be seen in this example that in the representation of "tall man" the variable Y is still uninstantiated. This variable will become instantiated after the ontology-based interpretation, performed by Φ_{onto}.

The semantic composition operation has four properties:

P1. Body variable specialization: $b = (b_1, \ldots, b_n)\nu$. This means that b is the concatenation of $b_1, \ldots b_n$ after the application of the ν substitution, which is a variable substitution $\{X_1/X, \ldots\}$ embedded in Φ_{comp} (see Figure 2). All variables in ν are also in h or $h_i, 1 \leq i \leq n$.

P2. Head constant generalization: $\Phi_{comp} = [(h \cup \bigcup_{1 \leq i \leq n} h_i)\mu]\nu$. This means that Φ_{comp} is the union of the semantic molecule heads to which a substitution μ for contextual generalization of constants $\{c_i/Y, \ldots\}$ is applied, followed by the substitution ν. The contextual substitution, μ, is specific to A_h, which is dependent on the nonterminal category [7]. The global substitution $\theta = \mu\nu$ is given as a subsystem of equations included in the constraint

7. Using the μ substitution for contextual constant generalization, the need of multiple examples for Φ_{comp} generalization can be reduced or even avoided (e.g., for cases such as syntactic agreement).

$\Phi_{comp}(h, h_1, \ldots, h_n)$, which is the system of equations given in (3a) and (3b). In the example presented in Figure 2 we have that:

$$\theta = \mu\nu = \left\{ \begin{array}{l} h.head = h_1.mod = X, \\ h.head = h_2.head = X, \\ h.nr = h_2.nr = X_4 \end{array} \right\}$$

P3. The Determinacy of the compositional semantic constraints: On the one hand, Φ_{comp} and the generalized syntagmas corresponding to the nonterminals from the right-hand side of the grammar rule, $(w_i, h_i \bowtie b_i)$, completely determine the generalized syntagma corresponding to the left-hand side nonterminal,$(w, h \bowtie b)$. On the other hand, $(w, h \bowtie b)$ and $(w_i, h_i \bowtie b_i)$ completely determine Φ_{comp}. The latter is relevant when learning Φ_{comp} together with the grammar rule. Besides, Φ_{comp} allows for parsing reversibility (see Section 3.3.3).

P4. Rule generalization: If $\Phi_{a'}, \Phi_b, \Phi_a$ are compositional constraints that obey the above three properties and use $\theta_{a'}, \theta_b, \theta_a$ as the corresponding substitutions, then the generalization rule: $\frac{A \to \alpha\ \beta\ \gamma\colon \Phi_{a'} \qquad B \to \beta\colon \Phi_b}{A \to \alpha\ B\ \gamma\colon \Phi_a}$, guarantees that $\theta_{a\prime} \subseteq \theta_b\theta_a$ and $L_\sigma(A \to \alpha\ B\gamma\colon \Phi_a) \supseteq L_\sigma(A \to \alpha\beta\gamma\colon \Phi_{a'})$.

3.2.2 Ontology-based Semantic Interpretation Constraint

The $\Phi_{onto}(b)$ constraint is applied only to the body of the semantic molecule corresponding to the left-hand side nonterminal, and provides an ontology-based semantic interpretation at the rule level. This constraint is used both during learning and during language analysis. It is built using a meta-interpreter with *freeze* (Saraswat, 1989) (Muresan, Potolea, & Muresan, 1998). This meta-interpreter assures that the atomic predicates, APs, (see Eq. (1)), of the molecule body are not evaluated (i.e., they are postponed) until at least one variable becomes instantiated. This technique allows a nondeterministic efficient search in the ontology. The meta-interpreter search strategy is independent of the actual representation of the ontology, and therefore behaves as an interface to any ontology at the level of atomic predicates. The ontology-based interpretation is not done during the composition operation, but afterwards. Thus, for example, the head of the noun phrase "tall man" (Figure 2) does not need to store the slot Y, a fact that allows us to use flat feature structures to represent the head of the semantic molecule. At this point, when Φ_{onto} is called, the variable Y becomes instantiated with values taken from the ontology (e.g., $height$). The ontology-based semantic interpretation constraint is important for the disambiguation required for some linguistic phenomena (e.g., prepositional phrase attachment, coordinations), and for semantic interpretation, including challenging phenomena, such as prepositions and noun-noun compounds.

A detailed description of the grammar constraints will be given in a forthcoming paper.

3.3 Lexicalized Well-Founded Grammars

Both in formal and linguistic theories of grammars, lexicalization is an important factor. Lexicalized grammars are finitely ambiguous and thus decidable (Joshi & Schabes, 1997). In our framework, we define a new type of constraint-based grammars, called *Lexicalized Well-Founded Grammars*, which augment the Well-Founded Grammars with semantics. A sublanguage of these grammars consists of pairs of strings and their semantic molecules, which we defined as *generalized syntagmas* ($\sigma = (w, w')$, see Definition 4).

Definition 5. A *Lexicalized Well-Founded Grammar (LWFG)* is a 6-tuple, $G = \langle \Sigma, \Sigma', N_G, R_G, P_G, S \rangle$, where:

(i) Σ is a finite set of terminal symbols.

(ii) Σ' is a finite set of elementary semantic molecules corresponding to the set of terminal symbols. That is, $w' \in \Sigma'$ iff $w \in \Sigma$, where $\sigma = (w, w')$.

(iii) N_G is a finite set of nonterminal symbols, $N_G \cap \Sigma = \emptyset$.

(iv) R_G is a partial ordering relation, $\succeq$, among the nonterminals.

(v) P_G is a set of constraint production rules. A constraint rule is a triple $(A, (B_1, \ldots, B_n), \Phi)$, written $A(\sigma) \rightarrow B_1(\sigma_1), \ldots, B_n(\sigma_n) \colon \Phi(\bar{\sigma})$, where $\bar{\sigma} = (\sigma, \sigma_1, ..., \sigma_n)$ such that $\sigma = (w, w'), \sigma_i = (w_i, w_i'), 1 \leq i \leq n, w = w_1 \cdots w_n, w' = w_1' \circ \cdots \circ w_n'$. Sometimes, for brevity, we denote a rule by $A \rightarrow \beta \colon \Phi$, where $\beta = B_1, \ldots, B_n$, and the arguments are variables[8]. For preterminals, we use either the $A(\sigma) \rightarrow$, or $A \rightarrow \sigma$ notation. These rules have the following properties: [9]

- There are three types of rules: ordered non-recursive rules, ordered recursive rules, and non-ordered rules.
- Every nonterminal symbol is a left-hand side in at least one ordered non-recursive rule.
- The empty string cannot be derived from any nonterminal symbol.
- The rule nonterminals are augmented with generalized syntagmas, σ, (i.e., pairs of strings and their semantic molecules).
- The rules are enriched with constraints, $\Phi(\bar{\sigma})$. There are two types of constraints: one for semantic composition and one for ontology-based semantic interpretation, as described in detail in Section 3.2.
- The rules (the representation and the constraints) assure grammar reversibility (see Section 3.3.3).

(vi) $S \in N_G$ is the start nonterminal symbol and $\forall A \in N_G, S \succeq A$.

(vii) In a Lexicalized Well-Founded Grammar all substrings w, derived from a nonterminal A have the same category of their semantic molecules, given by the name of the nonterminal. That is, $h.cat = A$, where $w' = h \bowtie b$ is the semantic molecule of w.

3.3.1 Derivation in LWFGs

Definition 6. Given a Lexicalized Well-Founded Grammar G, the *ground syntagma derivation*, '$\stackrel{*}{\Rightarrow}$' [10], is defined as: $\frac{A \rightarrow \sigma}{A \stackrel{*}{\Rightarrow} \sigma}$ (if $\sigma = (w, w'), w \in \Sigma, w' \in \Sigma'$, i.e. A is a preterminal), and

$$\frac{B_i \stackrel{*}{\Rightarrow} \sigma_i,\ i=1,...,n, \quad A(\sigma) \rightarrow B_1(\sigma_1),...,B_n(\sigma_n) \colon \Phi(\bar{\sigma}),\ \bar{\sigma}=(\sigma,\sigma_1,...,\sigma_n)}{A \stackrel{*}{\Rightarrow} \sigma}$$

8. When the arguments of the nonterminal are given, we understood them as being particular syntagmas attached to nonterminals.
9. The first three are the properties of the Well-Founded Grammars, while the last three are specific to the Lexicalized Well-Founded Grammars.
10. We use the notation $\stackrel{*G}{\Rightarrow}$ when the context requires the explicit mention of the grammar.

As can be noticed, in our framework, the grammar derives both the strings and their semantic molecules, i.e., the grammar derives generalized syntagmas. The ground syntagma derivation, $A \stackrel{*}{\Rightarrow} \sigma$, is equivalent to DCG provability, i.e., $P_G \vdash A(\sigma)$.

The *language* of a grammar G is the set of all syntagmas generated from the start symbol S, i.e., $L(G) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, S \stackrel{*}{\Rightarrow} \sigma\}$. The *set of all syntagmas* generated by *a grammar* G is $L_\sigma(G) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, \exists A \in N_G, A \stackrel{*}{\Rightarrow} \sigma\}$. For a grammar G, let E be a sublanguage, such that $E \subseteq L(G)$, and let $E_\sigma \subseteq L_\sigma(G)$ be the set of subsyntagmas corresponding to the sublanguage E. We have that $L(G) \subseteq L_\sigma(G)$ and $E \subseteq E_\sigma$ [11].

Extending the notation, given a grammar G, the set of syntagmas generated by *a nonterminal* A of the grammar G is $L_\sigma(A) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, A \in N_G, A \stackrel{*}{\Rightarrow} \sigma\}$, and the set of syntagmas generated by *a rule* $A \rightarrow \beta \colon \Phi$ of the grammar G is $L_\sigma(A \rightarrow \beta \colon \Phi) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, (A \rightarrow \beta \colon \Phi) \stackrel{*}{\Rightarrow} \sigma\}$[12].

3.3.2 SEMANTICS OF LWFGS

Operational Semantics. Following the insight of "parsing as deduction" (Shieber, Schabes, & Pereira, 1995), a deductive system for parsing Context-Free Grammars can serve as a method for defining their operational semantics. Moreover, it has been shown that the operational semantics of a CFG corresponds to the language of the grammar (Wintner, 1999). Analogously, in our framework, the operational semantics of a Lexicalized Well-Founded Grammar G is the set of all syntagmas generated by the grammar, $L_\sigma(G)$. That is $P_G \vdash A(\sigma)$ iff $\sigma \in L_\sigma(G)$.

Denotational Semantics. As discussed in literature (Pereira & Shieber, 1984; Wintner, 1999), the denotational semantics of a grammar is defined through a fixpoint of a transformational operator associated with the grammar.

Definition 7. Let $I \subseteq L_\sigma(G)$ be a subset of syntagmas generated by the grammar G. We define the *immediate syntagma derivation operator* $T_G \colon 2^{L_\sigma(G)} \rightarrow 2^{L_\sigma(G)}$, s.t.: $T_G(I) = \{\sigma \in L_\sigma(G) |$ if $(A(\sigma) \rightarrow B_1(\sigma_1), \ldots, B_n(\sigma_n) \colon \Phi(\bar{\sigma})) \in P_G \ \wedge \ B_i \stackrel{*}{\Rightarrow} \sigma_i \ \wedge \ \sigma_i \in I$ then $A \stackrel{*}{\Rightarrow} \sigma\}$. 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 $i = 1, T_G \uparrow 1 = T_G(\emptyset) = \{\sigma \in L_\sigma(G) | A \in N_G, A \rightarrow \sigma\}$. This corresponds to the syntagmas derived from preterminals, i.e., $\sigma = (w, w')$, where w' are elementary semantic molecules, $w' \in \Sigma'$.

T_G is analogous with the immediate consequence operator of definite logic programs (i.e., no negation) (van Emden & Kowalski, 1976; Denecker, Bruynooghe, & Marek, 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 & 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 syntagma derivation operator. An assumption for learning Lexicalized-Well Founded Grammars is that the rules corresponding to grammar preterminals are given: $A \rightarrow \sigma$, i.e., $T_G(\emptyset)$ is given (see assumption A3, Section 5.1).

11. In the remainder of this paper we will use the term *sublanguage* E_σ to refer to the set of subsyntagmas corresponding to the sublanguage E.
12. We use the notation $(A \rightarrow \beta \colon \Phi) \stackrel{*}{\Rightarrow} \sigma$ to denote the derivation $A \stackrel{*}{\Rightarrow} \sigma$ obtained using the rule $A \rightarrow \beta \colon \Phi$ in the last derivation step.

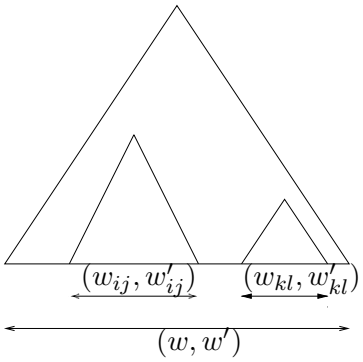


Figure 3: Robust parsing: returns all the chunks $(w_{ij}, w'_{ij}), (w_{kl}, w'_{kl})$ of a given syntagma $(w, w\prime)$.

As in the case of definite logic programs, the denotational semantics is equivalent with the operational one, i.e., $L_\sigma(G) = lfp(T_G)$.

Definition 8. Based on the immediate syntagma derivation operator, T_G, we can define the *ground derivation length (gdl)* for syntagmas, $gdl(\sigma)$, and the *minimum ground derivation length* for grammar rules, $mgdl(A \rightarrow \beta \colon \Phi)$:

$$gdl(\sigma) = \min_{\sigma \in T_G \uparrow i}(i)$$
$$mgdl(A \rightarrow \beta \colon \Phi) = \min_{\sigma \in L_\sigma(A \rightarrow \beta \colon \Phi)}(gdl(\sigma))$$

3.3.3 Reversible Robust Parsing of LWFGs

In our framework, the ground syntagma derivation is performed by a robust, bottom-up active chart parser (Kay, 1973). We use the term "robust" to convey that the parsing mechanism returns all the chunks, not only the full parses of a string (see Figure 3). The robust parser used in our framework is reversible. Informally, a reversible parser is a parser that can both parse (i.e., given a natural language string obtain its semantic representation) and generate (i.e., given the semantic representation obtain the natural language string). The semantics of our robust, bottom-up, reversible parser is given by the two definitions presented below.

Let's consider $(w, w') \in E_\sigma \subseteq L_\sigma(G)$ a syntagma derived by a grammar G, such that $w = w_1 \cdots w_n$ is a string, $w' = h \bowtie b$ is its semantic molecule, and $b = b_1 \cdots b_n$ is the string semantic representation.

Definition 9. We define the set of syntagmas *parsed* by the robust parser by:
$L_\sigma(w) = \{\sigma | \sigma = (w_{ij}, w'_{ij}),\ w_{ij} = w_i w_{i+1} \cdots w_j,\ 1 \leq i \leq j \leq n,\ \exists A \in N_G,\ A \stackrel{*}{\Rightarrow} \sigma\}$

Definition 10. We define the set of syntagmas *generated* by the robust parser by: $L_\sigma(b) = \{\sigma |$
$\sigma = (w_{ij},\ w'_{ij}),\ w'_{ij} = h_{ij} \bowtie b_{ij},\ b_{ij} = b_i b_{i+1} \cdots b_j,\ 1 \leq i \leq j \leq n,\ \exists A \in N_G,\ A \stackrel{*}{\Rightarrow} \sigma\}$

For all syntagmas, $\sigma \in L_\sigma(w)$, or $\sigma \in L_\sigma(b)$, the robust parser returns all rules $A \rightarrow \beta \colon \Phi$, such that $A \rightarrow \beta \colon \Phi \stackrel{*}{\Rightarrow} \sigma$, as well as the syntagma ground derivation length, $gdl(\sigma)$. In general, for a given syntagma $\sigma = (w, h \bowtie b)$ we may have that $L_\sigma(w) \neq L_\sigma(b)$, due to semantic ambiguity

(one string has many representations) or paraphrasing (many strings have the same representation), even for unambiguous LWFGs, defined in the next section (some examples are given in Appendix C).

4. Representative Examples

Any Lexicalized Well-Founded Grammar G induces a partial ordering on any generated sublanguage of *syntagmas*, $E_\sigma \subseteq L_\sigma(G)$. To show this, we need to introduce the notion of syntagma equivalence classes. The equivalence classes of syntagmas, eq_class(σ), are defined as pairs of nonterminals, (C, A). A lexicographic ordering, $\succeq_{lex}$, is assumed for (C, A) pairs. We add two symbols o and ρ to the set of nonterminals, N_G, i.e., $N_G^\dagger = N_G \cup \{o, \rho\}$, such that $o \prec \rho \prec A$, $\forall A \in N_G$. Also, we denote by $f_N(r \overset{*}{\Rightarrow} \sigma)$ the set of nonterminals that belong to the parse tree of $r \overset{*}{\Rightarrow} \sigma$, where $f_N \colon P_G \times E_\sigma \rightarrow N_G, r \in P_G, \sigma \in E_\sigma$ (see Figure 4).

Definition 11. A LWFG, G, is *unambiguous* if $\forall \sigma \in L_\sigma(G)$ there is one and only one rule $A \rightarrow \beta \colon \Phi \overset{*}{\Rightarrow} \sigma$ [13].

The equivalence classes of a sublanguage E_σ of an unambiguous LWFG, G, are computed by Algorithm 2.

Algorithm 2: Syntagma_Equivalence_Classes(E_σ, G)

foreach $(C_i, A_i) \in N_G \times N_G^\dagger$ **do**
  Equivalence_Class((C_i, A_i)) $\leftarrow \emptyset$
foreach $\sigma_i \in E_\sigma$ **do**
  $(k, gdl) \leftarrow$ eq_class (σ_i)
  Equivalence_Class(k) $\leftarrow$ Equivalence_Class(k) $\cup \{(\sigma_i, gdl)\}$
Topological_Sort(Equivalence_Class(k), E_{sort})
return E_{sort}

Procedure `eq_class(`σ_i`)`

$r \leftarrow (A_i \rightarrow \beta \colon \Phi)$ s.t. $A_i \rightarrow \beta \colon \Phi \overset{*}{\Rightarrow} \sigma_i$ /*given by the robust parser */
$i_\sigma \leftarrow gdl(\sigma_i)$ /*given by the robust parser */
$C_i \leftarrow$ Max_Nonterminal($f_N(r \overset{*}{\Rightarrow} \sigma_i) - \{A_i\}$) /*given by the robust parser */
1 **if** $A_i \succeq C_i$ *and* $C_i \not\succeq A_i$ **then** $k \leftarrow (A_i, o)$
else if $A_i \succeq C_i$ **then** $k \leftarrow (A_i, \rho)$
  else $k \leftarrow (C_i, A_i)$
2 $k \leftarrow \max\{k, \max_{\sigma_j \subset \sigma_i}(\text{eq\_class}(\sigma_j))\}$
return (k, i_σ)

For each syntagma $\sigma_i \in E_\sigma$ we choose the nonterminal A_i s.t. $A_i \rightarrow \beta \colon \Phi \overset{*}{\Rightarrow} \sigma_i$. The equivalence class of each subsyntagma $\sigma_j \subset \sigma_i$ is computed. Let C_i be the biggest nonterminal

13. Unambiguity is relative to syntagmas and not to language strings, which can be ambiguous. In the case of chains of unary branching rules the equivalent syntagmas of the same string must have different categories. More details in Section 5.1

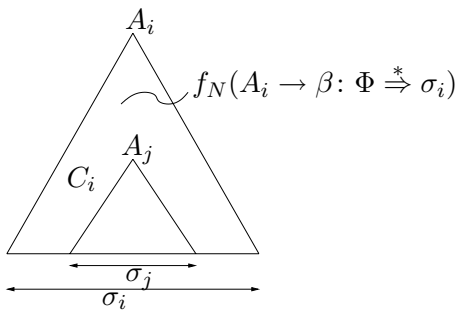


Figure 4: Syntagma partial ordering: $\sigma_i \succeq \sigma_j$.

of the parsing subtree rooted at A_i, except the root A_i (Figure 4). For all the rules for which σ_i is minimum, i.e, $gdl(\sigma_i) = mgdl(A_i \rightarrow \beta\colon \Phi)$, we have three cases: if the rule $A_i \rightarrow \beta\colon \Phi \stackrel{*}{\Rightarrow} \sigma_i$ is ordered, non-recursive, then $C_i \prec A_i$; if the rule is ordered, recursive, then $C_i = A_i$; and if the rule is non-ordered then $C_i \succ A_i$. The equivalence class of the minimum syntagma σ_i, eq_class(σ_i), is computed at step 1 as: (A_i, o) for an ordered, non-recursive rule; (A_i, ρ) for an ordered, recursive rule; and (C_i, A_i), for a non-ordered rule. For non-minimum syntagmas, the equivalence class may be changed at step 2.

Thus, the equivalence classes introduce a partial ordering relation among the syntagmas of the sublanguage: $\sigma_i \succeq \sigma_j$ iff $eq\_class(\sigma_i) \succeq_{lex} eq\_class(\sigma_j)$. Algorithm 2 returns the topologically sorted set E_{sort} of syntagmas σ_i, based on the partial ordering relation and the syntagma's ground derivation length, $gdl(\sigma_i)$ [14]: $\sigma_m \geq ... \geq \sigma_i \geq ... \geq \sigma_0$. The algorithm is polynomial in $|E_\sigma|$ and $|\sigma|$. The procedure eq_class is efficiently performed by a bottom-up active chart parser (Kay, 1973).

Lemma 2. *Algorithm 2 assures that any syntagma σ_i generated by an unambiguous LWFG, G, has the equivalence class greater than or equal with its subsyntagmas σ_j. That is $\sigma_i \succeq \sigma_j$ for all $\sigma_j \subset \sigma_i$. Moreover, in the totally ordered set E_{sort} returned by Algorithm 2, we have that $\sigma_i > \sigma_j$.*

Proof. The property $\sigma_i \succeq \sigma_j$ is guaranteed by the step 2 of Procedure eq_class, while property $\sigma_i > \sigma_j$ is guaranteed by the topological sorting of E_{sort}, where $gdl(\sigma_i) > gdl(\sigma_j)$. □

The topologically sorted set of syntagmas enables us to compute the representative examples of an unambiguous Lexicalized Well-Founded Grammar.

Definition 12. A set of syntagmas $E_R^G \subseteq L_\sigma(G)$ is called *representative example set* of an unambiguous LWFG, G, iff for each rule $(A \rightarrow \beta\colon \Phi) \in P_G$ there is a unique syntagma $\sigma \in E_R^G$ s.t. $gdl(\sigma) = mgdl(A \rightarrow \beta\colon \Phi)$.

From this definition it is straightforward that $|E_R^G| = |P_G|$. E_R^G contains the most simple syntagmas ground derived from the grammar G, and covers all the grammar rules.

Definition 13. Let G be an unambiguous Lexicalized Well-Founded Grammar. A sublanguage E_σ is called *complete* w.r.t the grammar G if it covers all grammar rules. That is, $\forall G^-$ with $P_{G^-} \subset P_G$,

14. Inside the same equivalence class the total ordering is done based on $gdl(\sigma_i)$. Moreover, if σ corresponds to the left-hand side nonterminal of a grammar rule, then $\sigma > \sigma_i, \forall \sigma_i$ on the right-hand side.

we have that $E_\sigma \subseteq L_\sigma(G) \wedge E_\sigma \not\subseteq L_\sigma(G^-)$. The grammar G is called the *minimal grammar* that covers E_σ.

The representative example set E_R^G of an unambiguous LWFG G is complete w.r.t the grammar G, because it covers all grammar rules. Given a sublanguage E_σ of an unambiguous LWFG G, such that $E_R^G \subseteq E_\sigma \subseteq L_\sigma(G)$, Algorithm 4 [15] computes iteratively the totally ordered set of grammar rules P_{G_r} from which E_σ ground derives, together with the totally ordered representative example set, $E_R^{G_r}$. At the beginning of each iteration, σ is the minimum syntagma of the totally ordered set $E_{sort} = E_\sigma$, which is still uncovered by the grammar G_r. The robust parser returns the unique rule r from which σ is ground derived, in the unambiguous grammar G. At the end of each iteration, G_r is enriched with r, while σ is added to $E_R^{G_r}$. The syntagmas covered by G_r at this point (including σ) are deleted from E_{sort}. The ordering of E_{sort} assures that $gdl(\sigma) = mgdl(r)$.

Algorithm 4: Find_Representative_Examples(E_σ, G)

$E_{sort} \leftarrow$ Syntagma_Equivalence_Classes(E_σ, G) /*$E_\sigma \subseteq L_\sigma(G)$ */
$E_R^{G_r} \leftarrow \emptyset$, $P_{G_r} \leftarrow \emptyset$
$k = 0$
repeat
  $k \leftarrow k + 1$
  $\sigma \leftarrow$ Extract_Min(E_{sort})
1 $r \leftarrow (A \rightarrow \beta\colon \Phi) \in P_G$ s.t. $A \rightarrow \beta\colon \Phi \stackrel{*G}{\Rightarrow} \sigma$ /*given by the robust parser */
2 $P_{G_r} \leftarrow P_{G_r} \cup \{(r, k)\}$
  $E_R^{G_r} \leftarrow E_R^{G_r} \cup \{(\sigma, k)\}$
  $E_{sort} \leftarrow E_{sort} - L_\sigma(G_r)$ /*$\sigma \in L_\sigma(G_r)$ */
until $E_{sort} = \emptyset$
return ($E_R^{G_r}$, P_{G_r})

Theorem 1 (Representative Examples Theorem). *Given an unambiguous Lexicalized Well-Founded Grammar G and a sublanguage E_σ s.t. $E_R^G \subseteq E_\sigma \subseteq L_\sigma(G)$, the Find_Representative_Examples algorithm generates in polynomial time the totally ordered representative example set, $E_R^{G_r}$, together with the associated totally ordered grammar rule set P_{G_r} that covers E_σ, such that $E_R^{G_r} = E_R^G$ and $P_{G_r} = P_G$. We use the notation $(G, E_\sigma) \xrightarrow{4} (E_R^{G_r}, G_r)$.*

Proof. At step 1 the following properties hold:

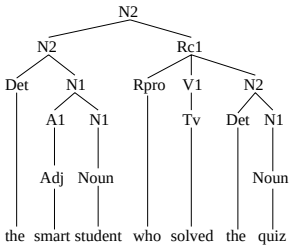
(i) $\exists! r \in P_G$ s.t. $r \stackrel{*G}{\Rightarrow} \sigma$ (since $\sigma \in E_\sigma \subseteq L_\sigma(G)$ and G is unambiguous)

(ii) $\sigma \notin L_\sigma(G_r)$ (since otherwise σ would have been previously deleted from E_{sort})

(iii) $\exists! r_i \in P_{G_r}$ s.t. $r_i \stackrel{*G_r}{\Rightarrow} \sigma_i$ for all $\sigma_i \subset \sigma$ (from Lemma 2 it follows that for all $\sigma_i \subset \sigma$, $\sigma_i < \sigma$ and thus they were previously deleted from E_{sort}. This implies that $\sigma_i \in L_\sigma(G_r)$)

(iv) $r \notin P_{Gr}$ (We assume the contrary. By (iii), if $r \in P_{G_r}$, it follows that $r \stackrel{*G_r}{\Rightarrow} \sigma$. This implies that $\sigma \in L_\sigma(G_r)$ which contradicts (ii))

15. Algorithm 4 can be also used for sublanguages E_σ that are not complete w.r.t. the grammar G. In this case, G_r, which is the minimal grammar that covers E_σ, is different from G.



(a)

$E_R^{G_r}$	Eq. class	P_{G_r}
smart	(A_1, o)	$A_1 \rightarrow Adj$
student	(N_1, o)	$N_1 \rightarrow Noun$
solved	(V_1, o)	$V_1 \rightarrow Tv$
smart student	(N_1, ρ)	$N_1 \rightarrow A_1\ N_1$
the quiz	(N_2, o)	$N_2 \rightarrow Det\ N_1$
who solved the quiz	(Rc_1, o)	$Rc_1 \rightarrow Rpro\ V_1\ N_2$
the student who solved the quiz	(Rc_1, N_2)	$N_2 \rightarrow N_2\ Rc_1$

(b)

Figure 5: (a) Parse tree ; (b) Results of Algorithm 4

In conclusion, at step 1, r is a new rule from P_G and thus σ is the minimum syntagma in E_{sort} such that $r \stackrel{*G}{\Rightarrow} \sigma$. This implies that $mgdl(r) = gdl(\sigma)$, because by hypothesis we have that $E_R^G \subseteq E_\sigma$. At step 2, P_{G_r} is enhanced with the new rule r and its index k, while $E_R^{G_r}$ is enhanced with the minimum syntagma σ that ground derives from it, and the same index k. As $r \in P_{G_r}$, it follows that $r \stackrel{*G_r}{\Rightarrow} \sigma$ (by property (iii)). Therefore, $\sigma \in L_\sigma(G_r)$ is deleted from E_{sort}. It follows that Algorithm 4 ends with $E_{sort} = \emptyset$, which implies that $E_\sigma \subseteq L_\sigma(G_r)$. The returned $E_R^{G_r}$ is the totally ordered representative example set of the minimal grammar G_r that covers the sublanguage E_σ, and P_{G_r} is the totally ordered set of grammar rules. Since E_σ is complete w.r.t. G it follows that $P_{G_r} = P_G$ and $E_R^{G_r} = E_R^G$. It is straightforward that the algorithm is polynomial in $|E_\sigma|$ and $|\sigma|$. □

The above theorem states that for a sublanguage E_σ complete w.r.t. an unambiguous LWFG G, such that $E_R^G \subseteq E_\sigma$, Algorithm 4 returns the totally ordered representative example set $E_R^{G_r}$ of the grammar G, together with the totally ordered grammar rule set P_{G_r} (this is a consequence of the well-foundness of the grammar nonterminal set). The small size of $E_R^{G_r}$ (i.e., equal to the size of the grammar) is an important feature, since the representative example set will be used as a semantic treebank for the grammar induction. Moreover, the total order that $E_R^{G_r}$ provides to the grammar rules is important since it allows a bottom-up induction of the grammar (see Section 5 and Table 1).

In the remainder of the paper we will use the notation E_R when the grammar is clearly understood from context, and for Algorithm 4 the notation $(G, E_\sigma) \stackrel{4}{\longrightarrow} (E_R, G)$.

Example. Figure 5(b) shows the results of Algorithm 4 given the sublanguage, E_σ, of the noun phrase "the smart student who solved the quiz", and the grammar G in Figure 1(a). Figure 5(a) shows the corresponding parse tree. For simplicity, we show only the strings without their semantic molecules.

5. Learnability of Lexicalized Well-Founded Grammars

In our previous work (Muresan et al., 2004), we stated the Grammar Induction Problem as follows: given a sublanguage $E_\sigma \subseteq L_\sigma(G)$ of an unknown Lexicalized Well-Founded Grammar G, together with its set of representative examples E_R, $E_R \subseteq E_\sigma$, learn a grammar G' such that $E_\sigma \subseteq L(G') \cap L(G)$. In this paper, we prove a much stronger theorem. We show that given certain assumptions for both the grammar G and the sublanguage E_σ we have that $G' = G$ (see Theorem 2).

5.1 Assumptions and Properties of Lexicalized Well-Founded Grammars

A 1. A property of Lexicalized Well-Founded Grammars (see Definition 5, (vii)) is that the category of a nonterminal is the name of the nonterminal: $\forall A \in N_G$ we have $h_A.cat = A$. As a consequence, for the unary branching rules, $A \rightarrow B\colon \Phi$, where $A, B \in N_G$, the syntagmas which are ground-derived from $A \rightarrow B\colon \Phi \stackrel{*}{\Rightarrow} \sigma_A$ and $B \stackrel{*}{\Rightarrow} \sigma_B$ have the same string w and the same semantic representation b, but have different valences $h_A \neq h_B$. Thus we can define the equivalence of two syntagmas and set of syntagmas:

(i) Two syntagmas $\sigma_1 = (w_1, h_1 \bowtie b_1)$ and $\sigma_2 = (w_2, h_2 \bowtie b_2)$ are equivalent, $\sigma_1 \equiv \sigma_2$, *iff* $w_1 = w_2 \wedge b_1 = b_2$

(ii) Two sets of syntagmas $L_{\sigma 1}$ and $L_{\sigma 2}$ are equivalent, $L_{\sigma 1} \equiv L_{\sigma 2}$, *iff* $(\forall \sigma_1 \in L_{\sigma 1}\ \exists \sigma_2 \in L_{\sigma 2}\ s.t.\ \sigma_1 \equiv \sigma_2) \wedge (\forall \sigma_2 \in L_{\sigma 2}\ \exists \sigma_1 \in L_{\sigma 1}\ s.t.\ \sigma_1 \equiv \sigma_2)$

Thus for unary branching rules, we have that $L_\sigma(A \rightarrow B\colon \Phi) \equiv L_\sigma(B)$ (i.e., they differ just by their valence, including their categories: $h_A.cat \neq h_B.cat$).

A 2. Considering the DCG-style formalism of Lexicalized Well-Founded Grammars, we assume that all the arguments of the nonterminals are variables, i.e., they are not instantiated with a particular value. This means that the right-hand side of all grammar rules cannot contain terminals (except for preterminals). This gives a syntactic overgeneralization, remaining to obtain a semantic specialization through the ontology-based interpretation, Φ_{onto}.

A 3. An assumption for learning Lexicalized-Well Founded Grammars is that the rules corresponding to the grammar preterminals are given: $A \rightarrow \sigma$, i.e., $T_G(\emptyset)$ is given (see denotational semantics, Section 3.3.2). This property imposes a refinement of the LWFG definition (Definition 5), requiring that $\beta \in N_G^+$, and not $\beta \in \{N_G \cup \Sigma\}^+$, for the rules that have nonterminals other than preterminals as their left-hand side.

As we saw in Section 4, we consider only unambiguous LWFGs. Two points should be made:

(i) Unambiguity refers to syntagmas and not to natural language expressions (strings). Two syntagmas $\sigma_1 = (w_1, h_1 \bowtie b_1)$ and $\sigma_2 = (w_2, h_2 \bowtie b_2)$ are equal, $(\sigma_1 = \sigma_2)$, *iff* $w_1 = w_2 \wedge h_1 = h_2 \wedge b_1 = b_2$. 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 (σ_1, σ_2 respectively), since $b_1 \neq b_2$ (in b_1 the PP is the adjunct of the verb "saw", while in b_2 the PP post-modifies the noun "man"). Thus, E_σ is unambiguous since σ_1 is derived from a single rule, and σ_2 is derived from another rule, even if the string corresponding to these two syntagmas is ambiguous. The same reasoning stands for the unary branching rules discussed above, since the syntagmas differ by their category (thus the semantic molecules associated with the strings differ by their heads this time). For examples the string "John" has two syntagmas associated with it: $\sigma_1 = (john, [cat : pn, head : X] \bowtie [X.name = john])$ and $\sigma_2 = (john, [cat : n, head : X] \bowtie [X.name = john])$, with $h_1 \neq h_2$, and thus $\sigma_1 \neq \sigma_2$. Thus, even if the string alone would be derived from two rules, one for PN, and another for N (we mentioned before that the category gives the name of the nonterminal), σ_1 is derived only from the rule corresponding to PN and σ_2 only from the rule corresponding to N (see also the examples given in Appendix C).

(ii) Unambiguity refers to syntagmas (a.k.a. representations) and not to interpreted natural language expressions. This means that a syntagma derived from a single rule can have many interpretations. For examples $(bone\ knife, [cat\colon n, ...] \bowtie [X_1.isa = bone, X_2.Y = X_1, X_2.isa = knife])$ has two interpretations (i.e., two values for the variable Y: "made of" and "purpose" given by Φ_{onto}). But it is unambiguous as representation, being derived from a single rule (noun compound rule in this case). In this paper we will not consider the interpretation ambiguity that is handled by the Φ_{onto} constraint, which nondeterministically returns all interpretations.

Definition 14. A Lexicalized Well-Founded Grammar G is *nonredundant* iff it does not contain equivalent nonterminals or rules, i.e., $A_i \neq A_j \ \ iff \ \ L_\sigma(A_i) \neq L_\sigma(A_j)$, and $A \rightarrow \beta_i\colon \Phi_i \neq A \rightarrow \beta_j\colon \Phi_j \ \ iff \ \ L_\sigma(A \rightarrow \beta_i\colon \Phi_i) \neq L_\sigma(A \rightarrow \beta_j\colon \Phi_j)$, respectively. [16]

Lemma 3. *An unambiguous LWFG G is nonredundant.*

Proof. The proof is immediate.

A key concept for proving the grammar learnability (see Sections 5.2 and 5.3) is the reduced grammar semantics defined below.

Definition 15. Given a LWFG, G and a sublanguage $E_\sigma \subseteq L_\sigma(G)$, we call $\mathbb{S}(G) = L_\sigma(G) \cap E_\sigma$ *the semantics of the grammar G reduced to the sublanguage E_σ*. Given a grammar rule $r \in P_G$, we call $\mathbb{S}(r) = L_\sigma(r) \cap E_\sigma$ *the semantics of the grammar rule r reduced to the sublanguage E_σ*.

Definition 16. A *chain* is a set of ordered unary branching rules: $\{B_k \rightarrow B_{k-1}, \ldots, B_2 \rightarrow B_1\}$ such that $L_\sigma(B_k) \supset L_\sigma(B_k \rightarrow B_{k-1}) \equiv L_\sigma(B_{k-1}) \supset \cdots \supset L_\sigma(B_2 \rightarrow B_1) \equiv L_\sigma(B_1)$ [17]. A *chain set* for a nonterminal B_i, $1 \leq i \leq k$, is $chs(B_i) = \{B_k, \ldots, B_1\}$. A grammar rule $r^+ = (A \rightarrow \beta^+\colon \Phi^+)$ is a *generalized rule* of a grammar rule $r = (A \rightarrow \beta\colon \Phi)$, if β^+ is formed by substituting a nonterminal B_i in β by a nonterminal ${B_i}^+$ if $\exists {B_i}^+ \in chs(B_i) \wedge {B_i}^+ \succ B_i$. A grammar rule $r^- = (A \rightarrow \beta^-\colon \Phi^-)$ is a *specialized rule* of a grammar rule $r = (A \rightarrow \beta\colon \Phi)$, if β^- is formed by substituting a nonterminal B_i in β by a nonterminal ${B_i}^-$ if $\exists {B_i}^- \in chs(B_i) \wedge {B_i}^- \prec B_i$. We call a LWFG, G, *general enough* w.r.t. a sublanguage E_σ, if for all generalized grammar rules we have $\mathbb{S}(r^+) = \mathbb{S}(r)$. We call a sublanguage E_σ *rich enough* w.r.t a LWFG, G, if for all specialized grammar rules we have $\mathbb{S}(r^-) \subset \mathbb{S}(r)$.

The general enough property of the grammar, and the rich enough property of the sublanguage used to reduce the grammar semantics, allow the rule generalization during grammar learning.

Definition 17 (Normalized). A Lexicalized Well-Founded Grammar G is called *normalized* (NLWFG) if for all grammar rules we have that $|\beta|$ is minim, i.e., $\forall A \rightarrow \beta\colon \Phi, \nexists$ a rule $A' \rightarrow \beta'\colon \Phi'$ with $(\beta' \subset \beta) \wedge (|\beta'| > 1)$.

The above mentioned definitions and assumptions allow us to introduce a new type of Normalized Lexicalized Well-Founded Grammar G that is *conform* to a sublanguage E_σ.

16. The grammar cannot contain unused rules since $S \succeq A, \forall A \in N_G$ (see Definition 5, (vi)). Also, every nonterminal symbol is a left-hand side in at least one rule (see Definition 5, (v)).
17. For simplicity, we use the notation $B_k \rightarrow B_{k-1}$ for the ordered unary branching rules $B_k(\sigma_k) \rightarrow B_{k-1}(\sigma_{k-1})\colon \Phi_k(\bar{\sigma})$, where $B_k \succ B_{k-1}$, and thus $\sigma_k \succ \sigma_{k-1}$ even if $\sigma_k \equiv \sigma_{k-1}$. We have that $\forall i, j, i \neq j, L_\sigma(B_i) \not\equiv L_\sigma(B_j)$.

Definition 18. A Normalized Lexicalized Well-Founded Grammar G is *conform* w.r.t. a sublanguage $E_\sigma \subseteq L_\sigma(G)$ iff G is unambiguous and general enough w.r.t. E_σ, and E_σ is complete and rich enough w.r.t. G.

Lemma 4. *Given a Normalized Lexicalized Well-Founded Grammar, G conform to E_σ, the order of nonterminals in any given chain maximizes the reduced grammar semantics $\mathbb{S}(G)$.*

Proof. Let's consider the chain $chs(B_1) = \{B_k, \ldots, B_1\}$, where $\forall i, j, 1 \leq i, j \leq k, i \neq j$ we have that $B_i \neq B_j$. If we switch nonterminals B_j and B_i, then grammar G becomes grammar G^*:

$G :: B_k \rightarrow \ldots \quad B_{j+1} \rightarrow \mathbf{B_j} \quad \mathbf{B_j} \rightarrow B_{j-1} \quad \ldots \quad B_{i+1} \rightarrow \mathbf{B_i} \quad \mathbf{B_i} \rightarrow B_{i-1} \ \ldots \ \rightarrow B_1$

$G^* :: B_k \rightarrow \ldots \quad B_{j+1} \rightarrow \mathbf{B_i} \quad \mathbf{B_i} \rightarrow B_{j-1} \quad \ldots \quad B_{i+1} \rightarrow \mathbf{B_j} \quad \mathbf{B_j} \rightarrow B_{i-1} \ \ldots \ \rightarrow B_1$

with the following change of rules:

	$G\ rule$		$G^*\ rule$
(4a)	$\begin{cases} B_{j+1} \rightarrow B_j \\ B_j \rightarrow B_{j-1} \end{cases}$	becomes specialized	$\begin{array}{r} B_{j+1} \rightarrow B_i \\ B_j \rightarrow B_{i-1} \end{array}$
(4b)	$\begin{cases} B_{i+1} \rightarrow B_i \\ B_i \rightarrow B_{i-1} \end{cases}$	becomes generalized	$\begin{array}{r} B_{i+1} \rightarrow B_j \\ B_i \rightarrow B_{j-1} \end{array}$

The first two rules (4a) are specialized rules and the last two rules (4b) are generalized rules, since $B_j \succ B_i$, and $B_{j-1} \succ B_{i-1}$. In the grammar $G \cup G^*$, for the first two rules we have that $\mathbb{S}(G^*\ rule) \subset \mathbb{S}(G\ rule)$ (E_σ is rich enough w.r.t. G), while for the last two rules we have that $\mathbb{S}(G^*\ rule) = \mathbb{S}(G\ rule)$ (G is general enough w.r.t E_σ). It follows that for the grammar G^* we have that $\mathbb{S}(G^*) \subset \mathbb{S}(G)$. This means that the original order maximizes the cardinality of the set $\mathbb{S}(G) = L_\sigma(G) \cap E_\sigma$. □

A 4. In order to prove the learnability theorem for grammar induction (see Section 5.3) we assume our target grammar to be a NLWFG conform w.r.t a sublanguage E_σ.

The NLWFGs are unambiguous, and thus, during the generation of the representative examples, E_R, of a grammar G, from a sublanguage E_σ, s.t. $E_R \subseteq E_\sigma \subseteq L_\sigma(G)$ (Algorithm 4, Theorem 1), each example σ_i has a unique rule r associated with it. Moreover, $mgdl(r) = gdl(\sigma_i)$, i.e., σ_i is the syntagma with the minimum ground derivation length, which is derived from r. Since the representative examples are ordered, this implies that they induce an order on the grammar rules. In Figure 6, the ith step of Algorithm 4 is given, which shows how the ith representative example is generated together with the rule from which it is derived. The rule $r = A_i \rightarrow \beta$: Φ is determined from P_G and σ_i, and then added to P_{G_r}. Since G is normalized and conform w.r.t. E_σ (assumption A4), it follows that the rule r cannot be further generalized. For all nonterminals $B_j \in \beta$ that belongs to the chain $chs(B_j)$, we have the following property: B_j is the minimum nonterminal in the chain $chs(B_j)$ that maximizes the reduced rule semantics $\mathbb{S}(r)$. This means that r is general enough w.r.t E_σ. The following must be noticed: given the assumptions A1, A2, A3, and A4, the ith rule, having the above property, can be generated, based only on the representative example σ_i,

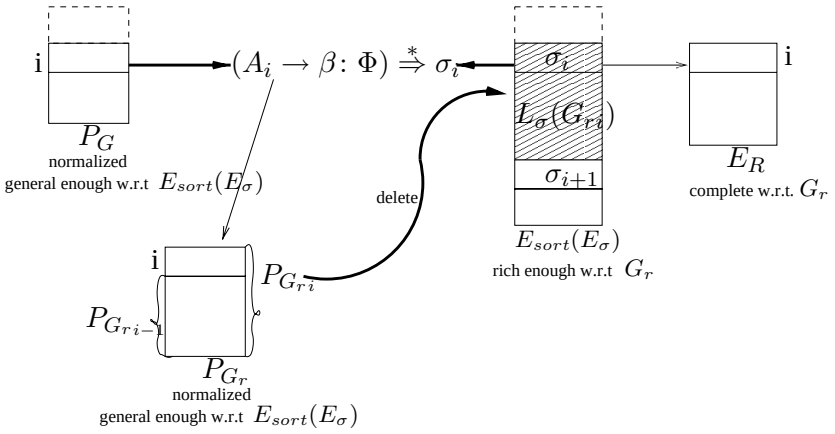


Figure 6: Step i in Algorithm 4. The lines in bold show that the rule associated with the minimum representative example σ_i is determined by P_G, and that σ_i belongs to $L_\sigma(G_{ri})$, which is deleted from the sublanguage E_σ.

the first $i-1$ rules of P_{G_r} and the sublanguage E_σ. This means that the grammar G_r can be learned bottom-up, i.e., the ith rule can be learned after the first $i-1$ rules are learned (Figure 8). The learning algorithm is presented in Section 5.2, and the learnability theorem in Section 5.3.

For a Normalized Lexicalized Well-Founded Grammar, G, conform to a set of syntagmas, E_σ, we have that for each equivalence class (C_i, A_i), the class (A_i, o), as well as the class (A_i, ρ) (when it exists) are generated using the Algorithm 2. Since $(A_i, o) \preceq (A_i, \rho) \preceq (C_i, A_i)$, for each nonterminal A_i, the learning algorithm will learn first the ordered non-recursive rules (shown in Figure 7), then the ordered recursive rules and last the non-ordered rules (see Algorithm 5). In the absence of this ordering, the learning machinery might need theory revision steps (see Algorithm 7).

Table 1 presents a summary of the main properties of Normalized Lexicalized Well-Founded Grammars and their consequences for learning. The first property, well-foundness of the grammar nonterminal set, allows for the total ordering of the grammar rule set, and thus a bottom-up induction of the grammar. This implication is shown in Figure 7, where it can be seen that the rules corresponding to preterminals are not learned (assumption A3), while all the other rules are learned bottom-up. The second and the third properties assure the termination condition for learning. The fourth property states that the category cat, given in the current representative example from which the rule is learned, provides the learner with the name of the predicate (i.e., the name of the left-hand side nonterminal). The fifth property shows which learning paradigm is suitable: Inductive Logic Programming based on Inverse Entailment (Muggleton, 1995), using as performance criterion the reduced grammar semantics, $\mathbb{S}(G)$. The sixth property allows us to efficiently learn complex rules (see Section 5.2, (Muresan et al., 2002)). Learning from a small number of examples has practical importance since semantic annotations are not readily available and are hard to build for a variety of domains. The last property allows us to learn only from positive data, which is essential, given that negative evidence is rarely available in language learning.

No	**Properties of NLWFGs**	**Consequences for Learning**
1	The set of grammar nonterminals is well founded	Bottom-up induction of the grammar
2	Every nonterminal is a left-hand side in at least one ordered non-recursive rule	Termination condition for induction [18]
3	The empty string ϵ cannot be derived from any nonterminal symbol	
4	All syntagmas σ, derived from a nonterminal *A*, have the same category of their semantic molecule (h.cat)	Predicate invention for induction
5	Ground syntagma derivation $A \stackrel{*}{\Rightarrow} \sigma$	Grammar-provability $K \wedge P_G \vdash A(\sigma)$
6	Ordered representative examples	Small semantic treebank for grammar induction
7	G conform to the sublanguage E_σ	Learnability only from positive examples

Table 1: Properties of NLWFGs and their implications for learning

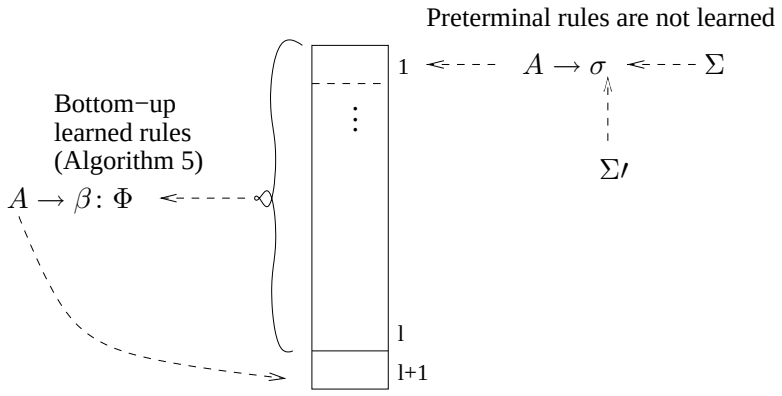


Figure 7: Nonterminal levels. Preterminals are on level 1 and their rules are not learned.

5.2 Relational Learning Algorithm

Most commonly, research in machine learning has focused on learning classification functions from data represented as vectors of attributes and their values (a.k.a, attribute-value representation). Even though this research has its own merits, there are more complex problems that require more expressive representations as well as the use of background knowledge during the learning process. Inductive Logic Programming (ILP), which is a class of relational learning, embodies both these characteristics (Muggleton & De Raedt, 1994; De Raedt, 1996; Lloyd, 2003). ILP methods have been used in a variety of applications for natural language processing(Cussens & Džeroski, 2000) and relational data mining, including applications for bioinformatics (Džeroski & Lavrač, 2001).

Our learning algorithm for grammar induction is based on our previous work (Muresan et al., 2002) and belongs to the class of Inductive Logic Programming methods (ILP), based on Inverse Entailment (Muggleton, 1995). Unlike existing relational learning methods that use randomly-

18. For all the rules, if σ corresponds to the left-hand nonterminal, then $\sigma > \sigma_i, \forall \sigma_i$ in the right-hand side, including chains (see footnote 14, 17).

selected examples and for which the class of efficiently learnable rules is very limited (Cohen, 1995), our algorithm learns from an ordered set of representative examples, allowing a polynomial efficiency for more complex rules. The size of this set is small and thus our algorithm is able to learn, where no large annotated treebanks can be easily built.

ILP methods have the ability to use background knowledge during learning. For our task, we use background knowledge K that contains: 1) the previously learned grammar, 2) the previously learned compositional semantic constraints, 3) the ontology, 4) the lexicon, which specifies for each word its part of speech, as well as the semantic information given as elementary semantic molecule, and 5) a reversible robust parser as innate inference engine.

The learning engine uses two sets of examples at different stages. First, the cover set algorithm is based only on the representative example set, E_R, which is semantically annotated (pairs of strings and their semantic molecules; see Appendix A and B for examples). During the generation of the final hypothesis, a second set E_σ is used for reducing the grammar semantics. The reduced grammar semantics is used in our Inverse Entailment learning method as the performance criterion in choosing the best rule. A characteristic of this set is that the examples can be just bracketed if this weakly annotation is enough to assure unambiguity. We denote these positive examples by E^+=E_σ.

Algorithm 5 describes the constraint-based grammar induction based only on positive examples. This algorithm is a refinement of our algorithm given in (Muresan et al., 2004), as a consequence of the refinement of the Lexicalized Well-Founded Grammars introduced in this paper (i.e., normalized, conform).

Algorithm 5: Constraint_Grammar_Induction(E_R, E_σ, K)

$P_{G'} \leftarrow \emptyset$
repeat
  $\sigma \leftarrow$ Extract_Min(E_R)
  $(A \rightarrow \beta\colon \Phi, Chs) \leftarrow$ Generate_Rule(σ, G', E_σ, K)
  $P_{G'} \leftarrow P_{G'} \cup \{A \rightarrow \beta\colon \Phi\}$
until $E_R = \emptyset$
return $P_{G'}$

Procedure `Generate_Rule`(σ, G', E_σ, K)

$\sigma = (w, h \bowtie b)$
1 $w \leftarrow \min(w_1 \dots w_n)$ s.t $b = (b_1, \dots, b_n)\nu$, where $w = w_1 \dots w_n$, $(w_j, h_j \bowtie b_j) \in L_\sigma(w)$, $1 \le j \le n$ /*n is minimum number of chunks; $L_\sigma(w)$ is generated by robust parsing */
2 $chs(j) = \{B_j | \sigma_j \equiv (w_j, h_j \bowtie b_j), B_j \stackrel{*}{\Rightarrow} \sigma_j\}, 1 \le j \le n$ /*by robust parsing */
3 $r_{max} \leftarrow (A \rightarrow {B_1}^+, \dots, {B_n}^+\colon \Phi^+)$ where ${B_j}^+ = \max(chs(j))$ with generalized arguments
$r \leftarrow (A \rightarrow B_1, \dots, B_n\colon \Phi)$, where $B_j = \min(chs(j))$ s.t. $\mathbb{S}(r) = \mathbb{S}(r_{max})$, and
$\Phi_{comp}(h, h_1, ..., h_n) = [(h \cup \bigcup_{1 \le i \le n} h_i)\mu]\nu$ /*determinacy property (P3) sec. 3.2.1 */
if $n > 1$ **then** return $(r, \emptyset)$
else return $(r, chs(1))$

For each representative example $\sigma \in E_R$, a cover set algorithm generates the corresponding rule (Generate_Rule) after which the rule together with the learned compositional semantic constraints are added to the background knowledge K, which contains the previously learned grammar G', and

the process continues iteratively until all the representative examples are covered. By the assumption A3, the rules corresponding to preterminals are not learned. They are generated from the lexicon and are given in the background knowledge K (see also Figure 7).

In step 1 of the Generate_Rule procedure, the robust parser generates the minimum number of chunks that cover σ (starting from the string w of σ) [19]. In step 2, for each chunk w_j , the robust parser determines $chs(j)$, i.e., the set of nonterminals from which $\sigma_j = (w_j, w'_j)$ is ground-derived (Figure 8). In step 3, the rule r_{max} is generated such that its left-hand side nonterminal is determined from the syntagma category, $h.cat = A$ (see property A1) and the arguments of each nonterminal B_j^+ from its right-hand side are generalized (see assumption A2). Then, the minimum $B_j \in chs(j)$ is chosen for the learned rule r, such that it has the same semantics as the rule r_{max} formed based on the maximum $B_j^+ \in chs(j)$ (see assumption A4). Therefore, in our Inverse Entailment learning method, the reduced grammar semantics is used as the performance criterion for selecting the final hypothesis r. The rule r_{max} is the most general rule, while r is the most specific rule that guarantees the same reduced semantics, $\mathbb{S}(r) = \mathbb{S}(r_{max})$. It is guaranteed that the rule r is normalized and general enough w.r.t. E_σ, in accord with assumption A4.

The algorithm is linear on the length of the learned hypothesis and has the complexity O($|E_R| * |\beta| * |chs(j)| * |E_\sigma| * |\sigma|^3$).

5.3 Learnability theorem

Theorem 2 (The NLWFG Induction Theorem). *Given a Normalized Lexicalized Well-Founded grammar, G, conform to a sublanguage E_σ, and a semantically annotated set $E_R \subseteq E_\sigma$ of ordered representative examples given by Algorithm 4: $(G, E_\sigma) \xrightarrow{4} (E_R, G)$, Algorithm 5 generates a grammar G' s.t. $G' = G$. We write $(E_R, E_\sigma) \xrightarrow{5} G$.*

Proof. Let's assume that after the first $i-1$ representative examples $\sigma_1, \ldots, \sigma_{i-1}$ we have that $P_{G'_{i-1}} = P_{G_{i-1}}$. In Algorithm 4, at step i, for σ_i we have that $A_i \rightarrow B_1, \ldots, B_n : \Phi \stackrel{*}{\Rightarrow} \sigma_i$ and the rule $A_i \rightarrow B_1, \ldots, B_n : \Phi$ is normalized and general enough w.r.t. E_σ (n is minimum (Figure 6)). Since G is conform to E_σ, it follows that Algorithm 5 (which guarantees that the learned rules are normalized and general enough w.r.t. E_σ), computes for σ_i, exactly the same rule $A_i \rightarrow B_1, \ldots, B_n : \Phi \stackrel{*}{\Rightarrow} \sigma_i$ at step i, and thus $P_{G'_i} = P_{G_i}$ (Figure 8). By complete induction, it follows that $G' = G$. □

5.4 Iterative Learning for Grammar Revision

Algorithm 5 presented in the previous section assumes a right order for the representative examples. However, in practice it might be difficult to provide the right order of examples, especially when modeling complex language phenomena. Algorithm 7 is an iterative grammar induction algorithm that starts with a random order of the representative example set, E_R^u. It scans all the representative examples (unordered), and for each example σ_i regenerates the rule r from which it can be derived, based on the current state of the other rules. For the unary branching rules (i.e., they belong to a chain, Chs), the position of the nonterminal associated with the representative example σ_i is

19. For σ with $gdl(\sigma) = mgdl(r)$, the chunks with the maximum length $|w_i|$ are efficiently computed by the robust parser, from left to right.

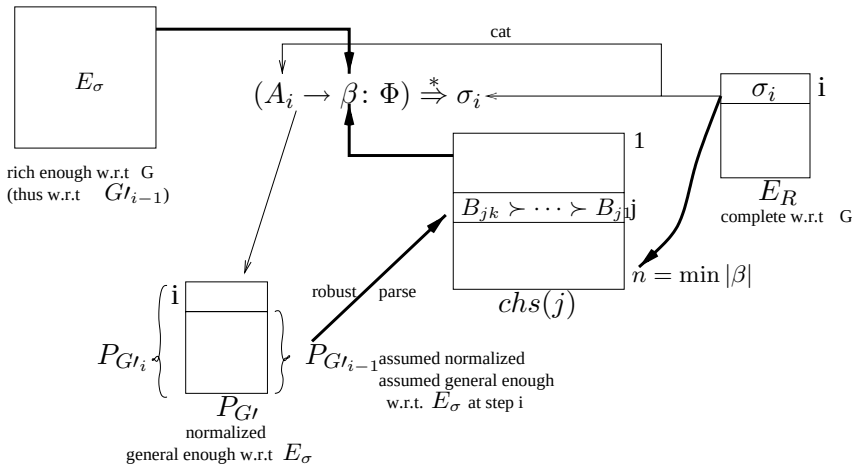


Figure 8: Step i in Algorithm 5. The lines in bold show that the rule rhs, β, is learned using the previous i-1 learned rules, $P_{G'_{i-1}}$, the current example, σ_i, and the sublanguage E_σ.

determined such that it maximizes the reduced semantics of the whole chain (Lemma 4). Thus the reduced semantics of all the rules/chains relative to E_σ is non-decreasing.

Algorithm 7: Iterative_Grammar_Induction(E_R^u, E_σ, K)

$P_{G'} \leftarrow \emptyset$
repeat
  $O_{G'} \leftarrow P_{G'}$
  for $i \leftarrow 1$ ***to*** $|E_R^u|$ **do**
    $\sigma_i \leftarrow E_R^u(i)$ /*unordered set of representative examples */
    $r = (A \rightarrow \beta\colon \Phi)$ s.t $\sigma_i \in L_\sigma(r)$
    if $|\beta| > 1$ **then** $P_{G'} \leftarrow P_{G'} - \{r\}$
    $(r, Chs) \leftarrow$ Generate_Rule($\sigma_i, G', E_\sigma, K$) /*regenerates the ith rule based on all the other rules */
    if $Chs = \emptyset$ **then** $P_{G'} \leftarrow P_{G'} \cup \{r\}$
    else Maximize(r, Chs) /*maximizes the reduced semantics of the chain, Chs */
until $O_{G'} = P_{G'}$
return $P_{G'}$

Theorem 3. *Given a Normalized Lexicalized Well-Founded Grammar G conform w.r.t. a sublanguage E_σ, and a semantic annotated set $E_R^u \subseteq E_\sigma$ of representative examples in random order, Algorithm 7 learns the same grammar as Algorithm 5 would do, if provided with the representative example set E_R in the right order. We write $(E_R^u, E_\sigma) \stackrel{7}{\longrightarrow} G$ iff $(E_R, E_\sigma) \stackrel{5}{\longrightarrow} G$.*

Proof. Let $L_i^k = \mathbb{S}(r)$ be the semantics of the ith grammar rule, r, reduced to E_σ, at iteration step k. From the property P4 of semantic composition (see Section 3.2.1), and Lemma 4 (see Section 5.1), we have that $L_i^k \supseteq L_i^{k-1}$, for $1 \leq i \leq |E_R^u|$. This implies that L_i^k converges (it is non-decreasing and bounded). Let G be the grammar obtained as limit by Algorithm 7, i.e., $(E_R^u, E_\sigma) \stackrel{7}{\longrightarrow} G$. The

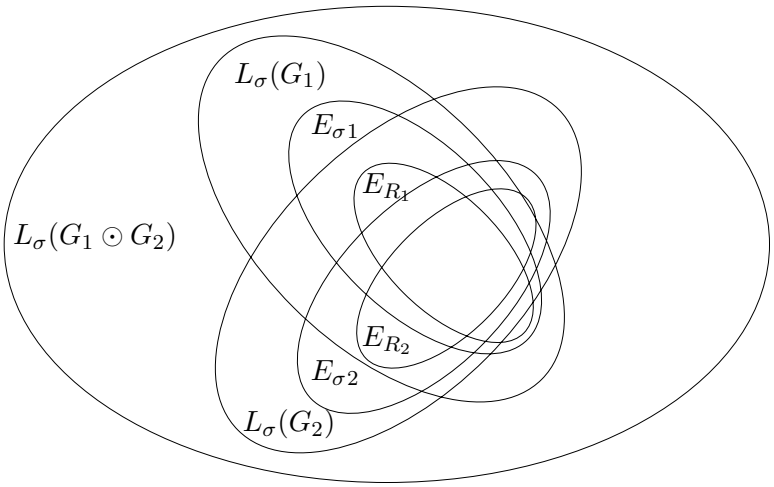


Figure 9: Merging two NLWFGs G_1 and G_2

grammar G is conform to E_σ [20], which implies that $(G, E_\sigma) \xrightarrow{4} (E_R, G)$ (Theorem 1), and thus $(E_R, E_\sigma) \xrightarrow{5} G$ (Theorem 2). Proving the reciprocal is immediate, since it is sufficient to take E_R^u in the right order in Algorithm 7.

6. Merging Normalized Lexicalized Well-Founded Grammars

In the previous sections, it could be noticed that for a Normalized Well-Founded Grammar G conform to a sublanguage E_σ, Algorithm 4 and Algorithms 5/7 allow for the reciprocal generation of the grammar rules P_G and the representative examples E_R mediated by the sublanguage E_σ. We denote this reciprocal generation $G \xleftrightarrow{E_\sigma} E_R$. The direction $\rightarrow$ is given by Algorithm 4 and the direction $\leftarrow$ by Algorithm 5/7, respectively.

Definition 19 (Grammar merging). Let G_1 and G_2 be two Normalized Lexicalized Well-Founded Grammars defined on two sets of nonterminals, N_{G_1} and N_{G_2}, respectively, such that $N_{G_1} \subseteq N_G$, $N_{G_2} \subseteq N_G$

$$G_1 = \langle \Sigma, \Sigma\prime, N_{G_1}, R_{G_1}, P_{G_1}, S_1 \rangle$$
$$G_2 = \langle \Sigma, \Sigma\prime, N_{G_2}, R_{G_2}, P_{G_2}, S_2 \rangle$$

where R_{G_1} and R_{G_2} are consistent with each other [21]. The subset of nonterminals (other than preterminals), which are common to N_{G_1} and N_{G_2}, is called the *cut nonterminal set*. Let $E_{\sigma 1}$ and $E_{\sigma 2}$ be the sublanguages corresponding to the grammar G_1 and G_2, respectively.

The merging of the grammars G_1 and G_2 is realized in three steps :

(i) From G_1, G_2 and the sublanguages $E_{\sigma 1}, E_{\sigma 2}$, Algorithm 4 is used to generate the sets of representative examples corresponding to these grammars, E_{R_1} and E_{R_2} respectively.

20. The Generate_Rule procedure guarantees that the learned rules are normalized and general enough w.r.t. E_σ, while the Maximize procedure guarantees the maximum reduced semantics for the chains.

21. Between the common nonterminals there is no contradictory partial ordering relation.

Using chains of nonterminals	Using adequate attributes for category definition
$A \rightarrow \alpha\mathbf{B}\gamma\colon \Phi_a$	$A \rightarrow \alpha\mathbf{B}\gamma\colon \Phi_a$
$\mathbf{B} \rightarrow \beta_1\colon \Phi_1$	$\mathbf{B} \rightarrow \beta_1\colon \Phi_1 \quad (h.a = v12)$
$\mathbf{B} \rightarrow \beta_2\colon \Phi_2$	$\mathbf{B} \rightarrow \beta_2\colon \Phi_2 \quad (h.a = v12)$
$B^+ \rightarrow B\colon \Phi^+$	$B \rightarrow \beta_3\colon \Phi_3 \quad (h.a = v3)$
$B^+ \rightarrow \beta_3\colon \Phi_3$	

Figure 10: Two ways of overcoming overgeneralization. The rules with the nonterminal B in bold as their left-hand side are the only ones allowed in the ground derivation of the rule $A \rightarrow \alpha B\gamma\colon \Phi_a$. This is obtained by introducing a new nonterminal B^+, or a discriminative attribute $h.a$.

(ii) The union of the sets of representative examples, $E_{R_1} \cup E_{R_2}$ [22], and the union of the sublanguages, $E_{\sigma 1} \cup E_{\sigma 2}$ are performed.

(iii) Algorithm 5 or Algorithm 7 are applied to these two sets, obtaining the merged grammar $G = G_1 \odot G_2$.

That is, if :

$$G_1 \overset{E_{\sigma 1}}{\longleftrightarrow} E_{R_1}$$
$$G_2 \overset{E_{\sigma 2}}{\longleftrightarrow} E_{R_2}$$

then:

$$G_1 \odot G_2 \overset{E_{\sigma 1} \cup E_{\sigma 2}}{\longleftrightarrow} E_{R_1} \cup E_{R_2}$$

Theorem 4. *Merging two Normalized Lexicalized Well-Founded Grammars G_1 and G_2 assures that: $L_\sigma(G_1) \cup L_\sigma(G_2) \subseteq L_\sigma(G_1 \odot G_2)$ (see Figure 9).*

Proof. The proof is immediate. □

If the nonterminals belonging to the cut nonterminal set have different semantics in the two grammars, then we have that $L_\sigma(G_1) \cup L_\sigma(G_2) \subset L_\sigma(G_1 \odot G_2)$ (strict subset relation). Thus, in general, this merging method can lead to overgeneralization. However, the overgeneralization can be avoided by using chains of nonterminals or by introducing adequate attributes to define meaningful categories (i.e., nonterminals). An abstract example of using these two methods to avoid overgeneralization is given in Figure 10. It can be seen that the first method implies introducing an additional nonterminal B^+, while the second method uses the same nonterminal B, but with

22. E_{R_1} and E_{R_2} allow the automatic alignment of the nonterminals of the two grammars (including the ones belonging to the same chain). For this, the equivalence of syntagmas is exploited $\sigma_1 \equiv \sigma_2$, where $b_1 = b_2$ and $h_1.cat \neq h_2.cat$. The alignment determines the cut nonterminal set. During merging, Algorithm 7 could iterate only the rules that corresponds to nonterminals that belong to the cut nonterminal set, increasing the efficiency.

	$\boldsymbol{G_1}$	$\boldsymbol{G_2}$	$\boldsymbol{G = G_1 \odot G_2}$
P_G	$N_2 \to Det\ Noun \colon \Phi_4$ $N_2 \to Pn \colon \Phi_5$ $C_1 \to N_2\ Tv\ N_2 \colon \Phi_6$	$N_1 \to Noun \colon \Phi_1$ $N_1 \to Adj\ N_1 \colon \Phi_2$ $N_2 \to Det\ N_1 \colon \Phi_3$	$N_1 \to Noun \colon \Phi_1$ $N_1 \to Adj\ N_1 \colon \Phi_2$ $N_2 \to Det\ N_1 \colon \Phi_3$ $N_2 \to Pn \colon \Phi_5$ $C_1 \to N_2\ Tv\ N_2 \colon \Phi_6$
Σ	$Noun \to [child]$ $Noun \to [day]$ $Adj \to [sunny]$ $Adj \to [playful]$ $Tv \to [like]$ $Det \to [the]$ $Det \to [a]$ $Pn \to [john]$	$Noun \to [child]$ $Noun \to [day]$ $Adj \to [sunny]$ $Adj \to [playful]$ $Tv \to [like]$ $Det \to [the]$ $Det \to [a]$ $Pn \to [john]$	$Noun \to [child]$ $Noun \to [day]$ $Adj \to [sunny]$ $Adj \to [playful]$ $Tv \to [like]$ $Det \to [the]$ $Det \to [a]$ $Pn \to [john]$
L_σ	{ *the child*, ***john***, ***the child likes john***, ... }	{ ***sunny day***, *the child*, ***the playful child***, ... }	{ *sunny day*, *the child*, *the playful child*, *john*, *the child likes john*, ***john likes a sunny day***, ***the playful child likes a sunny day*** ... }

(a)

$\boldsymbol{E_{\sigma 1}}$	$\boldsymbol{E_{\sigma 2}}$	$\boldsymbol{E_{\sigma 1} \cup E_{\sigma 2}}$
	child	child
	playful child	playful child
	nice playful child	nice playful child
the child	the child	the child
	the playful child	the playful child
john		john
john likes the child		john likes the child
the child likes john		the child likes john

(b)

$\boldsymbol{E_{R_1}}$	$\boldsymbol{E_{R_2}}$	$\boldsymbol{E_{R_1} \cup E_{R_2}}$
	child	child
	playful child	playful child
the child	the child	the child
john		john
john likes the child		john likes the child

(c)

Figure 11: Merging two grammars G_1 and G_2

Nr. Rules $\lvert P_G\rvert = \lvert E_R\rvert$	$\lvert N_G\rvert$ *nonterminals*	*NL fragment learned*	*Number of rules involving*				
			NP	*Aux*	*Verb*	*Core Cl*	*Wh-clause*
13	6	Aux + subj agr.	2	11			
26	8	verb (iv,tv,dv) + subj agr.			26		
12	4	core clause + wh-clause	3			6	3
25	10	complex NP + noun-compound	25				
76	28	*Total*					

Figure 12: Statistics of a learned grammar

a discriminative attribute $h.a$. This attribute helps discriminate between the first two alternatives and the third one (by having two different values: $v12$ and $v3$, where only $v12$ is accepted by the constraint Φ_a).

In Figure 11, we show an example of merging two grammars (G_1 and G_2). $\{N_2\}$ is the cut nonterminal set. In this example, it can be seen that merging two grammars is not the union of their production rules, i.e., $P_{G_1 \odot G_2} \neq P_{G_1} \cup P_{G_2}$. The grammar G_1 generates only simple sentences (e.g., "the child likes John"), while the grammar G_2 generates more complex noun phrases (e.g., modified by adjectives: "sunny day", "the playful child"). The merged grammar G generates more complex sentences: "the playful child likes a sunny day". While this is just an illustrative example, one can imagine a real case where a grammar that generates complex sentences is obtained by merging several grammars (e.g., simple clauses, complex noun phrases, complex verb constructions with auxiliaries). In Section 7, we present experiments done in this direction. The merging method presented in this section shows that we model the grammar learning/development from simple to complex (which is a cognitively plausible approach, simulating the child language acquisition process (Pinker, 1989)).

7. Linguistic Relevance and Experimental Results

In this section we discuss the linguistic relevance of our Normalized Lexicalized Well-Founded Grammars, by presenting several grammatical aspects that can be modeled as well as the application of the learning methodology presented in the previous sections. Our grammars can model aspects relevant to linguistic theories and of practical use in Natural Language Processing applications (e.g., noun-noun compounds, prepositional phrases, coordinations). [23]

We follow a functional approach to syntax, similar with (van Valin & LaPolla, 1997). Figure 12 presents statistics of applying our learning framework to several fragments of natural language: 1) Auxiliary verb constructions including subject agreement, 2) Verb constructions for intransitive, transitive and ditransitive verbs, including subject agreement, 3) Core clause (verb + its arguments) and wh-clauses (interrogatives and relative clauses), and 4) Complex noun phrases (with determiners and adjectival premodifiers, with prepositional phrases, genitive constructions, coordination, and noun compounds).

23. The experiments presented in this paper are given only to illustrate the applicability of the theoretical concepts. More experimental results and a thorough evaluation will be provided in a paper where the theoretical aspects will not be the focus.

The learned grammar contains 76 rules and 76 compositional semantic constraints. We used 13 preterminals (i.e., POS), 23 elementary semantic molecules $\in \Sigma'$, 28 nonterminal names (other than preterminals) and 76 representative examples, E_R. An important thing to notice is that the number of representative examples is the same as the number of the learned rules (see Figure 12).

During learning, an additional set of 236 examples E_σ is used for reducing the grammar semantics during rule generalization. A characteristic of this set is that it does not need to be fully annotated if a simple bracketing is enough for disambiguation. This bracketing can be seen as "dummy semantics" (Culicover & Nowak, 2003). The full annotation is used only when the bracketing cannot disambiguate (e.g., for auxiliary constructions).

Consider G_{aux} the learned grammar for auxiliaries, G_{vb} the learned grammar for verbal constructions, G_{clwh} the learned grammar for core clauses and wh-clauses, and G_{np} for complex noun phrases and noun compounds. In Appendix A and B we give the representative examples, the learned grammars and sample of learned compositional constraints for G_{aux} and a fragment of G_{np}. We briefly describe below the phenomena covering the verbal and noun phrase constructions.

Learning auxiliary constructions. In this experiment we focused on modeling complex auxiliary forms that appear in complex finite verbs constructions. The learned grammar G_{aux} deals with modals, negation, subject-verb agreement, tense and aspect. We give the learned grammar below. It can be noticed that we introduced the subject at this level in order to facilitate agreement and subject-auxiliary inversion that appears in questions ($AV0$). In this grammar we only have pronouns and proper names as subject. More complex subject constructions will result after grammar merging. In the learned grammar, we have 4 nonterminals for auxiliaries. For example, $AV0$ models simple form of auxiliaries "be" and "have" as well as modal auxiliaries and the periphrastic auxiliary "do", together with subject agreement and inversion; $AV1$ is used for further modeling constructions with relative pronouns used either in questions or relative clause constructions (it can be noticed that in this case we do not have inversion); $AV2$ introduces negation; $AV3$ introduces the future tense and the perfect aspect; while $AV4$ introduces the progressive form of the auxiliary "to be", which will be used in conjunction with the passive constructions (e.g "*she may have been being* examined by ..."). In this grammar we have the following chain of nonterminals $\{AV4, AV3, AV2, AV1, AV0\}$.

Learned Grammar, G_{aux}
$Sbj(h \bowtie b) \rightarrow Pro(h_1 \bowtie b_1)$: $\Phi_{comp1}(h, h_1), \Phi_{onto}(b)$
$Sbj(h \bowtie b) \rightarrow PN(h_1 \bowtie b_1)$: $\Phi_{comp2}(h, h_1), \Phi_{onto}(b)$
$AV0(h \bowtie b) \rightarrow Sbj(h_1 \bowtie b_1),\ Aux(h_2 \bowtie b_2)$: $\Phi_{comp3}(h, h_1, h_2), \Phi_{onto}(b)$
$AV0(h \bowtie b) \rightarrow Aux(h_1 \bowtie b_1),\ Sbj(h_2 \bowtie b_2)$: $\Phi_{comp4}(h, h_1, h_2), \Phi_{onto}(b)$
$AV1(h \bowtie b) \rightarrow AV0(h_1 \bowtie b_1)$: $\Phi_{comp5}(h, h_1), \Phi_{onto}(b)$
$AV1(h \bowtie b) \rightarrow RelPro(h_1 \bowtie b_1),\ Aux(h_2 \bowtie b_2)$: $\Phi_{comp6}(h, h_1, h_2), \Phi_{onto}(b)$
$AV2(h \bowtie b) \rightarrow AV1(h_1 \bowtie b_1)$: $\Phi_{comp7}(h, h_1), \Phi_{onto}(b)$
$AV2(h \bowtie b) \rightarrow AV1(h_1 \bowtie b_1),\ Aux(h_2 \bowtie b_2)$: $\Phi_{comp8}(h, h_1, h_2), \Phi_{onto}(b)$
$AV3(h \bowtie b) \rightarrow AV2(h_1 \bowtie b_1)$: $\Phi_{comp9}(h, h_1), \Phi_{onto}(b))$
$AV3(h \bowtie b) \rightarrow AV2(h_1 \bowtie b_1),\ Aux(h_2 \bowtie b_2)$: $\Phi_{comp10}(h, h_1, h_2), \Phi_{onto}(b)$
$AV3(h \bowtie b) \rightarrow AV3(h_1 \bowtie b_1),\ Aux(h_2 \bowtie b_2)$: $\Phi_{comp11}(h, h_1, h_2), \Phi_{onto}(b)$
$AV4(h \bowtie b) \rightarrow AV3(h_1 \bowtie b_1)$: $\Phi_{comp12}(h, h_1), \Phi_{onto}(b)$
$AV4(h \bowtie b) \rightarrow AV3(h_1 \bowtie b_1),\ Aux(h_2 \bowtie b_2)$: $\Phi_{comp13}(h, h_1, h_2), \Phi_{onto}(b)$

Sample of the learned compositional constraints	
$\Phi_{comp2}(h, h_1)$ =	$\{h$.cat=sbj, h.dets=y, h.pers=h_1.pers, h.nr=h_1.nr, h.case=h_1.case, h.hum=h_1.hum, h.head=h_1.head, h_1.cat=pn, h_1.det=y$\}$
$\Phi_{comp8}(h, h_1, h_2)$ =	$\{h$.cat=av2, h.int=h_1.int, h.dets=h_1.dets, h.case=h_1.case, h.hum=h_1.hum, h.aux=h_1.aux, h.neg=h_2.neg, h.tense=h_1.tense, h.pers=h_1.pers, h.nr=h_1.nr, h.pf=h_1.pf, h.pg=h_1.pg, h.headS=h_1.headS, h.head=h_1.head, h.head=h_2.head, h_1.cat=av1, h_1.neg=no, h_2.cat=aux, h_2.aux=not$\}$
$\Phi_{comp11}(h, h_1, h_2)$ =	$\{h$.cat=av3, h.int=h_1.int, h.dets=h_1.dets, h.case=h_1.case, h.hum=h_1.hum, h.aux=h_2.aux, h.neg=h_1.neg, h.tense=h_1.tense, h.pers=h_1.pers, h.nr=h_1.nr, h.pf=h_2.pf, h.pg=h_1.pg, h.headS=h_1.headS, h.head=h_1.head, h.head=h_2.head, h_1.cat=av3, h_1.aux=have, h_1.pf=no, h_2.cat=aux, h_2.vf=en$\}$
$\Phi_{comp13}(h, h_1, h_2)$ =	$\{h$.cat=av4, h.int=h_1.int, h.dets=h_1.dets, h.case=h_1.case, h.hum=h_1.hum, h.aux=h_1.aux, h.aux=h_2.aux, h.neg=h_1.neg, h.tense=h_1.tense, h.pers=h_1.pers, h.nr=h_1.nr, h.pf=h_1.pf, h.pg=h_2.pg, h.headS=h_1.headS, h.head=h_1.head, h.head=h_2.head, h_1.cat=av3, h_1.pg=no, h_2.cat=aux, h_2.vf=ing$\}$

For learning this grammar we used a set of 13 fully annotated representative examples (see Appendix A) as well as a set of 39 additional examples E_σ which were also fully annotated because of the high lexical ambiguity of "have" and "be", which can have the same lexical form for several syntactic constructions (e.g., the same lexical item for 1st and 2nd person singular "I have" and "You have"). At this point the ontology cannot provide any disambiguation. Even if the rules of the grammar seem simple, and someone might wonder if they could be written by hand, the complexity of the task is emphasized in the learned constraints. An example of parsing and generation using this grammar is given in Appendix C.

Learning verbal constructions. In this experiment we modeled simple and complex finite verbal constructions for intransitive, transitive and ditransitive verbs, covering subject verb agreement, active and passive constructions, tense and aspect information. We used a set of 26 representative examples, E_R, 76 additional examples that constitute the sublanguage E_σ and G_{aux} as background knowledge. While tense and aspect information are represented in our grammar, we have not yet developed the interpretation module to account for temporal relations. This is part of future work.

Learning complex noun phrases. In this experiment we modeled several phenomena of complex noun phrases: prepositional phrases, coordinations, noun-noun compounds, genitive constructions. We used 25 representative examples and 85 additional examples that constitute the sublanguage E_σ. Part of this additional set of examples is just bracketed. As we mentioned before, this is a form of dummy semantics which is sufficient for disambiguation. Thus, we do not have to rely on fully annotated data for the whole set E_σ. This is important since the size of E_σ is larger than E_R. We give a fragment of our learned grammar G_{np} and the selected compositional constraints (in Appendix B we also give the representative examples from which this fragment was learned). In this grammar the noun compounds are given by the rules corresponding to the nonterminals Na and Nc, where Na generates constructions where nouns behave like adjectives and could be further combined with another noun to form a full-fledged noun compound. For example "skin disease treatment" can be a full-formed noun compound (generated by Nc) or can be further combined with the noun "effect" to obtain "skin disease treatment effect". In this case, it is generated by Na. It can be noticed that the learned rule for Na is both left and right recursive. In Appendix C we can see this phenomenon by showing a parsing/generation of the noun "skin disease treatment". It can also be noticed that the ontology-based semantic interpretation filters spurious parses. Thus, even though we obtain overgeneralization in syntax, the semantic interpretation at the rule level removes the

wrong parses. In order to control overgeneralization, besides using the chain $\{Sbj, N, Nc, Noun\}$, we also used discriminative attributes for the nonterminal N. For example, the difference between a determinate and a nondeterminate noun is given by the attribute det, which takes two values: yes and no respectively (otherwise we should have used two nonterminals $N1$ and $N2$). In this case, we can also see the complexity of the learned compositional constraints. For example Φ_{comp9} will constrain the rule $N \rightarrow Det\ N$, and thus it will not generate recursively ungrammatical phrases such as "the the disease". This is obtain by having the constraints $h.det = y, h_2.det = no$, which say that the value of the attribute det for the noun on the left-hand side of the rule is yes, while for the noun on the right-hand side is no. Currently we ignore individuals at the interpretation level. We do not semantically represent plurals, while determiners even if represented, are not interpreted [24]. In our future work we plan to include the interpretation of individuals.

Fragment of Learned Grammar, G_{np}
$A(h \bowtie b) \rightarrow Adj(h_1 \bowtie b_1)$: $\Phi_{comp1}(h, h_1), \Phi_{onto}(b)$
$A(h \bowtie b) \rightarrow Adv(h_1 \bowtie b_1),\ A(h_2 \bowtie b_2)$: $\Phi_{comp2}(h, h_1, h_2), \Phi_{onto}(b)$
$Na(h \bowtie b) \rightarrow Noun(h_1 \bowtie b_1)$: $\Phi_{comp3}(h, h_1), \Phi_{onto}(b)$
$Na(h \bowtie b) \rightarrow Na(h_1 \bowtie b_1),\ Na(h_2 \bowtie b_2)$: $\Phi_{comp4}(h, h_1, h_2), \Phi_{onto}(b)$
$Nc(h \bowtie b) \rightarrow Noun(h_1 \bowtie b_1)$: $\Phi_{comp5}(h, h_1), \Phi_{onto}(b)$
$Nc(h \bowtie b) \rightarrow Na(h_1 \bowtie b_1),\ Nc(h_2 \bowtie b_2)$: $\Phi_{comp6}(h, h_1, h_2), \Phi_{onto}(b)$
$N(h \bowtie b) \rightarrow Nc(h_1 \bowtie b_1)$: $\Phi_{comp7}(h, h_1), \Phi_{onto}(b)$
$N(h \bowtie b) \rightarrow A(h_1 \bowtie b_1),\ N(h_2 \bowtie b_2)$: $\Phi_{comp8}(h, h_1, h_2), \Phi_{onto}(b)$
$N(h \bowtie b) \rightarrow Det(h_1 \bowtie b_1),\ N(h_2 \bowtie b_2)$: $\Phi_{comp9}(h, h_1, h_2), \Phi_{onto}(b)$
$N(h \bowtie b) \rightarrow PN(h_1 \bowtie b_1)$: $\Phi_{comp10}(h, h_1), \Phi_{onto}(b)$
$N(h \bowtie b) \rightarrow Pro(h_1 \bowtie b_1)$: $\Phi_{comp11}(h, h_1), \Phi_{onto}(b)$
$Sbj(h \bowtie b) \rightarrow N(h_1 \bowtie b_1)$: $\Phi_{comp12}(h, h_1), \Phi_{onto}(b)$

Sample of learned compositional constraints	
$\Phi_{comp4}(h, h_1, h_2)$ =	$\{h$.cat=na, h.head=h_1.mod, h.head=h_2.head, h.mod=h_2.mod, h_1.cat=na, h_2.cat=na$\}$
$\Phi_{comp7}(h, h_1)$ =	$\{h$.cat=n, h.det=no, h.pers=h_1.pers, h.nr=h_1.nr, h.case=h_1.case, h.hum=h_1.hum, h.gen=h_1.gen, h.count=h_1.count, h.head=h_1.head, h_1.cat=nc, h_1.det=no$\}$
$\Phi_{comp9}(h, h_1, h_2)$ =	$\{h$.cat=n, h.det=y, h.pers=h_2.pers, h.nr=h_2.nr, h.case=h_2.case, h.hum=h_2.hum, h.gen=h_2.gen, h.count=h_2.count, h.head=h_1.head, h.head=h_2.head, h_1.cat=det, h_2.cat=n, h_2.det=no$\}$

In the experiments of learning these grammars, we have studied all the theoretical aspects presented in the previous sections. First, the grammar G_{aux} was learned using Algorithm 5 , when the order of examples was good (see Appendix A) , and Algorithm 7, when we scrambled the order of examples. In both cases, we obtained the same grammar. The resulting grammar has 13 rules and 13 learned compositional constraints. Samples of these constraints are given in Appendix A. Then, we applied the learning algorithm using G_{aux} as background knowledge to obtain the grammar G_{vb}. After that, by using $G_{aux} + G_{vb}$ as background knowledge we learned G_{chwh}. This incremental learning has the practical advantage of a better running time for learning the grammars and of a modular development of grammars. Finally, the resulted grammar $G_{aux} + G_{vb} + G_{clwh}$ and the grammar G_{np} learned separately were merged. During merging, the cut nonterminal set is $\{Sbj, Obj\}$ (see Appendix A and B), which means that the rules corresponding to this nonterminal set will be the

24. Our current practical application of the grammar is to build a terminological knowledge base, where there is no need for individuals.

ones revised (replaced by rules that have the maximum reduced semantics). This learning method followed by grammar merging allows us to learn from simple up to complex constructions.

The grammar learning framework presented in this paper is a valuable tool for grammar development, since it allows the refinement of the grammar by refining just the examples. Providing the representative examples can be challenging for some more complex phenomena. We can test whether they were properly chosen: first we learn the grammar from these examples E_R and the sublanguage E_σ using Algorithm 5 / 7. Then we regenerate the representative examples from the learned grammar and the sublanguage E_σ, using Algorithm 4. If the resulting set is not the same, it means that the examples were not adequate and might need to be revised. A way of validating a grammar and ultimately the derived semantic constructions, consists of using a reversible parser for parsing and generation. In Appendix C we give examples of parsing/generation for syntagmas corresponding to G_{aux}, G_{np} and G_{clwh}. It can be noticed that our robust parser is nondeterministic, giving all valid solutions. However, as can be seen for the noun compound "skin disease treatment", the ontological constraint filters some of the parses that are not validated by the semantic interpretation (in this case the attachment of the noun "skin" to the noun "treatment"). Thus, some of the syntactic overgeneralizations are specialized by semantic constraints.

The results presented in this section show promise for the linguistic adequacy of our proposed new type of grammars. Moreover, the theoretical foundations of the learnability and merging make the framework appealing for developing natural language grammars.

8. Conclusions

In this paper we have presented a new type of constraint-based grammars, *Normalized Lexicalized Well-Founded Grammars (NLWFGs)*. Motivated by the need of learnable grammars for language understanding, NLWFGs have been proved to have both good computational and learning properties, as well as relevant linguistic characteristics.

We have shown that for NLWFGs conform to a sublanguage E_σ, this sublanguage mediates the reciprocal generation of the grammar rules and the representative examples E_R. The representative examples are the simplest examples that can be derived from the grammar and consist of ordered pairs of natural language strings and their *semantic molecules*. The semantic molecule is a new representation for natural language strings, which contains information for semantic composition and encodes the string semantic representation as an Ontology Query Language. We have presented the algorithms and the associated theorems for generating the representative examples, E_R from a grammar G and a sublanguage E_σ (Theorem 1), as well as for learning grammar G, from the set of representative examples and the sublanguage E_σ used to reduce the grammar semantics (Theorem 2). Thus, the representative example set can be seen as a small semantically annotated treebank. From a theoretical perspective, the ordering of examples allows for an efficient bottom-up relational learning of complex constraint-based grammar rules and their compositional semantic constraints. However, providing the right order of examples might be difficult in practice, when modeling complex phenomena. We have presented an iterative learning algorithm for grammar revision that guarantees to obtain the same normalized grammar, regardless of the order of examples (Theorem 3).

Besides the learnability results, we have presented a principle for grammar merging based on the union of their representative examples and a subsequent application of the learning algorithm (Theorem 4). We have shown that grammar merging is not the same as the union of their rules.

All the theoretical aspects have been embedded in a relational learning framework for inducing NLWFGs and we have presented experimental results for modeling several aspects of natural language. One of the main conclusions of this paper is that if a fragment of natural language can be covered by a NLWFG G, and the semantically annotated representative examples E_R, as well as the sublanguage E_σ are provided based on linguistic knowledge, then the grammar G can be learned. This implies that if natural language can be covered by NLWFGs, and there is linguistic knowledge to build E_R and E_σ, then natural language can be learned. In our experiments we have shown that a fragment and several aspects of natural language can be covered by NLWFGs, and relatively simple linguistic knowledge is required to build the set of representative examples.

Further work is needed to use these grammars for broad coverage of natural language. An important direction would be to develop the framework to allow the bootstrapping of both the grammar and the ontology and to add probabilities at the ontology level.

References

Baker, C., Fillmore, C., & Lowe, J. (1998). The Berkeley FrameNet project. In *Proceedings of COLING/ACL*, pp. 86–90, Montreal, Canada.

Cardie, C. (1997). Empirical methods in information extraction. *AI Magazine*, *18*(4), 65–79.

Carpenter, B., & Penn, G. (1999). ALE: The Attribute Logic Engine, User's Guide. Tech. rep., Lucent Technologies and Universtat Tubingen.

Carreras, X., & Marquez, L. (2004). Introduction to CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of The Eight Conference on Natural Language Learning*.

Chen, J., & Rambow, O. (2003). Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of EMNLP 2003*.

Cohen, W. (1995). Pac-learning recursive logic programs: Negative results. *Journal of Artificial Intelligence Research*, *2*, 541–573.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Copestake, A. (1999). The (new) LKB system. Tech. rep., Stanford University.

Copestake, A., Flickinger, D., & Sag, I. A. (1999). Minimal Recursion Semantics: An introduction..

Copestake, A., Lascarides, A., & Flickinger, D. (2001). An algebra for semantic construction in constraint-based grammars. In *Proceedings of ACL-2001*.

Culicover, P., & Nowak, A. (2003). *Dynamical Grammar: Minimalism, Acquisition, and Change*. Oxford University Press.

Cussens, J., & Džeroski, S. (Eds.). (2000). *Learning Language in Logic, volume 1925 of Lecture Notes in Computer Science*. Springer-Verlag.

Dalrymple, M. (Ed.). (1999). *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. MIT Press, Cambridge, MA.

De Raedt, L. (Ed.). (1996). *Advances in Inductive Logic Programming*. IOS Press, Amsterdam, The Netherlands.

Denecker, M., Bruynooghe, M., & Marek, V. W. (2001). Logic programming revisited: Logic programs as inductive definitions. *ACM Transactions on Computational Logic*, *2*(4), 623–654.

Dudau-Sofronie, Tellier, I., & Tommasi (2001). Learning Categorial Grammars from semantic types. In *Proceedings of the 13th Amsterdam Colloquium*, pp. 79–84.

Džeroski, S., & Lavrač, N. (Eds.). (2001). *Relational Data Mining*. Springer, Berlin, Germany.

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering*, *6*(1), 15–28.

Gildea, D., & Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*, *28*(3).

Jones, R., Ghani, R., Mitchell, T., & Riloff, E. (2003). Active learning for information extraction with multiple view feature sets. In *Proceeding of ECML-03 Workshop on Adaptive Text Extraction and Mining*.

Joshi, A., & Schabes, Y. (1997). Tree-Adjoining Grammars. In Rozenberg, G., & Salomaa, A. (Eds.), *Handbook of Formal Languages*, Vol. 3, chap. 2, pp. 69–124. Springer, Berlin,New York.

Kay, M. (1973). The MIND system. In Rustin, R. (Ed.), *Natural Language Processing*, pp. 155–188. Algorithmics Press, New York.

Kingsbury, P., Palmer, M., & Marcus, M. (2002). Adding semantic annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference*, San Diego, California.

Klein, D., & Manning, C. D. (2001). Natural language grammar induction using a constituent-context model. In Dietterich, T. G., Becker, S., & Ghahramani, Z. (Eds.), *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Lloyd, J. (2003). *Logic for Learning: Learning Comprehensible Theories from Structured Data*. Springer, Cognitive Technologies Series.

Macherey, K., Och, F., & Ney, H. (2001). Natural language understanding using statistical machine translation. In *Proceedings of EuroSpeech '01*.

Miller, S., Bobrow, R., Ingria, R., & Schwartz, R. (1994). Hidden understanding models of natural language. In *Proceedings of ACL'94*.

Muggleton, S., & De Raedt, L. (1994). Inductive Logic Programming:Theory and methods. *Journal of Logic Programming*, *19*(20), 629–679.

Muggleton, S. (1995). Inverse Entailment and Progol. *New Generation Computing, Special Issue on Inductive Logic Programming*, *13*(3-4), 245–286.

Muresan, S. (2004). Inducing constraint-based grammars using a domain ontology. In *Proceedings of the Ninth AAAI/SIGART Doctoral Consortium*, San Jose, California.

Muresan, S., Muresan, T., & Klavans, J. (2004). Inducing constraint-based grammars from a small semantic treebank. In *Proceedings of AAAI Spring Symposium on Language Learning: An Interdisciplinary Perspective*, Stanford University.

Muresan, S., Muresan, T., & Potolea, R. (2002). Data flow coherence constraints for pruning the search space in ILP tools. *International Journal of Artificial Intelligence Tools*, *11*(2).

Muresan, T., Potolea, R., & Muresan, S. (1998). Amalgamating CCP with Prolog. *Scientific Journal of Politechnics University, Timisoara*, *43*(4).

Oepen, S., Flickinger, D., Toutanova, K., & Manning, C. D. (2002). LinGO Redwoods: A rich and dynamic treebank for HPSG. In *Proceedings of The First Workshop on Treebanks and Linguistic Theories (TLT2002)*.

Osborne, M. (1999). DCG induction using MDL and parsed corpora. In Cussens, J. (Ed.), *Proceedings of the 1st Workshop on Learning Language in Logic*, pp. 63–71, Bled, Slovenia.

Paroubek, P., Schabes, Y., & Joshi, A. K. (1992). XTAG: a graphical workbench for developing tree-adjoining grammars. In *Proceedings of the third conference on Applied natural language processing*, pp. 223–230.

Pereira, F. C., & Shieber, S. M. (1984). The semantics of grammar formalisms seen as computer languages. In *Proceeding of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association of Computational Linguistics*, pp. 123–129.

Pereira, F. C., & Warren, D. (1980). Definite Clause Grammars for language analysis. *Artificial Intelligence*, *13*, 231–278.

Pinker, S. (1989). *Learnability and Cognition: The Acquisition of Argument Structure*. MIT Press.

Pollard, C., & Sag, I. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, Illinois.

Retore, C., & Bonato, R. (2001). Learning rigid lambek grammars and minimalist grammars from structured sentences. In *Proceedings of the Third Learning Language in Logic Workshop*.

Saraswat, V. (1989). *Concurrent Constraint Programming Languages.* Ph.D. thesis, Dept. of Computer Science, Carnegie Mellon University.

Shieber, S., Schabes, Y., & Pereira, F. (1995). Principles and implementation of deductive parsing. *Journal of Logic Programming*, *24*(1-2), 3–36.

Siskind, J. (2000). Learning word-to-meaning mappings. In Broeder, P., & Murre, J. (Eds.), *Models of Language Acquisition: Inductive and Deductive Approaches*. Oxford University Press.

Starkie, B. (2002). Inferring attribute grammars with structured data for natural language processing. In *ICGI '02: Proceedings of the 6th International Colloquium on Grammatical Inference*, pp. 237–248. Springer-Verlag.

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

Thompson, C., & Mooney, R. (2003). Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research*, *18*, 1–44.

van Emden, M., & Kowalski, R. (1976). The semantics of predicate logic as a programming language. *Journal of the ACM*, *4*, 733–742.

van Valin, R. D., & LaPolla, R. J. (1997). *Syntax : Structure, Meaning, and Function*. Cambridge University Press.

Wintner, S. (1999). Compositional semantics for linguistic formalisms. In *Proceedings of the Association for Computational Linguistics, ACL'99*.

Wintner, S. (2002). Modular Context-Free Grammars. *Grammars*, *5*, 41–63.

Appendix A. Learning Auxiliary Verb Constructions

We present below the set of representative examples E_R required to learn the grammar. The set size is $|E_R| = 13$.

Representative Examples Set, $E_R = (w, w')$, where $w' = h \bowtie b$	
([he],	[cat=sbj, dets=y, pers=3, nr=sg, case=n, hum=y, head=X]⋈[X.isa=he]).
([john],	[cat=sbj, dets=y, pers=3, nr=sg, case=n, hum=y, head=X]⋈[X.name=john]).
([someone, is],	[cat=av0, int=no, dets=y, case=n, hum=_, aux=be, neg=no, tense=pr, pers=3, nr=sg, pf=no, pg=no, headS=X, head=Y]⋈[X.isa=someone, Y.tense=pr]).
([is, someone],	[cat=av0, int=y, dets=y, case=n, hum=_, aux=be, neg=no, tense=pr, pers=3, nr=sg, pf=no, pg=no, headS=X, head=Y] ⋈ [Y.tense=pr, X.isa=someone]).
([is, he],	[cat=av1, int=y, dets=y, case=n, hum=y, aux=be, neg=no, tense=pr, pers=3, nr=sg, pf=no, pg=no, headS=X, head=Y]⋈[Y.tense=pr, X.isa=he]).
([who, is],	[cat=av1, int=_, dets=no, case=n, hum=y, aux=be, neg=no, tense=pr, pers=3, nr=sg, pf=no, pg=no, headS=X, head=Y]⋈[X.isa=who, Y.tense=pr]).
([who, is],	[cat=av2, int=_, dets=no, case=n, hum=y, aux=be, neg=no, tense=pr, pers=3, nr=sg, pf=no, pg=no, headS=X, head=Y]⋈[X.isa=who, Y.tense=pr]).
([someone, is, not],	[cat=av2, int=no, dets=y, case=n, hum=_, aux=be, neg=y, tense=pr, pers=3, nr=sg, pf=no, pg=no, headS=X, head=Y]⋈[X.isa=someone, Y.tense=pr, Y.neg=y]).
([someone, is, not],	[cat=av3, int=no, dets=y, case=n, hum=_, aux=be, neg=y, tense=pr, pers=3, nr=sg, pf=no, pg=no, headS=X, head=Y]⋈[X.isa=someone, Y.tense=pr, Y.neg=y]).
([someone, will, be],	[cat=av3, int=no, dets=y, case=n, hum=_, aux=be, neg=no, tense=mod, pers=3, nr=sg, pf=no, pg=no, headS=X, head=Y]⋈[X.isa=someone, Y.mod=will, Y.tense=mod]).
([she, has, not, been],	[cat=av3, int=no, dets=y, case=n, hum=y, aux=be, neg=y, tense=pr, pers=3, nr=sg, pf=y, pg=no, headS=X, head=Y]⋈[X.isa=she, Y.tense=pr, Y.neg=y, Y.pf=y]).
([she, has, not, been] [25],	[cat=av4, int=no, dets=y, case=n, hum=y, aux=be, neg=y, tense=pr, pers=3, nr=sg, pf=y, pg=no, headS=X, head=Y]⋈[X.isa=she, Y.tense=pr, Y.neg=y, Y.pf=y]).
([someone, is, not, being],	[cat=av4, int=no, dets=y, case=n, hum=_, aux=be, neg=y, tense=pr, pers=3, nr=sg, pf=no, pg=y, headS=X, head=Y]⋈[X.isa=someone, Y.tense=pr, Y.neg=y, Y.pg=y]).

A set of 39 additional examples, $E^+ = E_\sigma$, fully annotated were used as a sublanguage for the rule generalization process. Below we show the learned grammar G_{aux} and the compositional semantic constraints. The learned grammar has 13 rules, $|P_G| = 13$. Thus $|E_R| = |P_G|$.

25. In order to reduce the size of E_σ needed for rule generalization we used in this experiment representative examples that do not have minimum derivation length.

Learned Grammar, G_{aux}. The rules are given in the DCG formalism[26]
sbj(H⋈B-Bo)−− >pro(H1⋈B-Bo), { fi-comp1(H,H1),fi-onto(B-Bo)}.
sbj(H⋈B-Bo)−− >pn(H1⋈B-Bo),{fi-comp2(H,H1),fi-onto(B-Bo)}.
av0(H⋈B-Bo)−− >sbj(H1⋈B-B1),aux(H2⋈B1-Bo),{fi-comp3(H,H1,H2),fi-onto(B-Bo)}.
av0(H⋈B-Bo)−− >aux(H1⋈B-B1),sbj(H2⋈B1-Bo),{fi-comp4(H,H1,H2),fi-onto(B-Bo)}.
av1(H⋈B-Bo)−− >av0(H1⋈B-Bo),{fi-comp5(H,H1),fi-onto(B-Bo)}.
av1(H⋈B-Bo)−− >relpro(H1⋈B-B1),aux(H2⋈B1-Bo),{fi-comp6(H,H1,H2),fi-onto(B-Bo)}.
av2(H⋈B-Bo)−− >av1(H1⋈B-Bo),{fi-comp7(H,H1),fi-onto(B-Bo)}.
av2(H⋈B-Bo)−− >av1(H1⋈B-B1),aux(H2⋈B1-Bo),{fi-comp8(H,H1,H2),fi-onto(B-Bo)}.
av3(H⋈B-Bo)−− >av2(H1⋈B-Bo),{fi-comp9(H,H1,fi-onto(B-Bo))}.
av3(H⋈B-Bo)−− >av2(H1⋈B-B1),aux(H2⋈B1-Bo),{fi-comp10(H,H1,H2),fi-onto(B-Bo)}.
av3(H⋈B-Bo)−− >av3(H1⋈B-B1),aux(H2⋈B1-Bo),{fi-comp11(H,H1,H2),fi-onto(B-Bo)}.
av4(H⋈B-Bo)−− >av3(H1⋈B-Bo),{fi-comp12(H,H1),fi-onto(B-Bo)}.
av4(H⋈B-Bo)−− >av3(H1⋈B-B1),aux(H2⋈B1-Bo),{fi-comp13(H,H1,H2),fi-onto(B-Bo)}.

Sample of the learned compositional constraints[27]	
fi-comp2(h, h1) =	{h.cat=sbj, h.dets=y, h.pers=h1.pers, h.nr=h1.nr, h.case=h1.case, h.hum=h1.hum, h.head=h1.head, h1.cat=pn, h1.det=y}
fi-comp8(h, h1, h2) =	{h.cat=av2, h.int=h1.int, h.dets=h1.dets, h.case=h1.case, h.hum=h1.hum, h.aux=h1.aux, h.neg=h2.neg, h.tense=h1.tense, h.pers=h1.pers, h.nr=h1.nr, h.pf=h1.pf, h.pg=h1.pg, h.headS=h1.headS, h.head=h1.head, h.head=h2.head, h1.cat=av1, h1.neg=no, h2.cat=aux, h2.aux=not}
fi-comp11(h, h1, h2) =	{h.cat=av3, h.int=h1.int, h.dets=h1.dets, h.case=h1.case, h.hum=h1.hum, h.aux=h2.aux, h.neg=h1.neg, h.tense=h1.tense, h.pers=h1.pers, h.nr=h1.nr, h.pf=h2.pf, h.pg=h1.pg, h.headS=h1.headS, h.head=h1.head, h.head=h2.head, h1.cat=av3, h1.aux=have, h1.pf=no, h2.cat=aux, h2.vf=en}
fi-comp13(h, h1, h2) =	{h.cat=av4, h.int=h1.int, h.dets=h1.dets, h.case=h1.case, h.hum=h1.hum, h.aux=h1.aux, h.aux=h2.aux, h.neg=h1.neg, h.tense=h1.tense, h.pers=h1.pers, h.nr=h1.nr, h.pf=h1.pf, h.pg=h2.pg, h.headS=h1.headS, h.head=h1.head, h.head=h2.head, h1.cat=av3, h1.pg=no, h2.cat=aux, h2.vf=ing}

Appendix B. Learning Fragments of Noun Phrases

In this section we give examples of fragments of NP covering noun-noun compounds.

26. In the DCG formalism, the nonterminal names begin with lower-case letters (as Prolog predicate names) and the logical variable names begin with capital letters. The strings w are implicit arguments in DCG, and thus not shown. $B-Bo$ is the notation used for difference lists applied to the semantic representations (i.e., the bodies of the semantic molecules) attached to each nonterminals. The constraints are enclosed in {}.
27. Given in syntactic sugaring notation used throughout the paper for better understanding. fi-comp are Prolog predicates, where the arguments are variables, as can be seen in the grammar given in the DCG form.

Representative Examples Set, $E_R = (w, w')$, where $w' = h \bowtie b$	
([young],	[cat=a, head=X, mod=Y] $\bowtie$ [X.isa=young, Y.P=X]).
([very, young],	[cat=a, head=X, mod=Y] $\bowtie$ [X.how=very, X.isa=young, Y.P=X]).
([disease],	[cat=na, head=X, mod=Y] $\bowtie$ [X.isa=disease, Y.P=X]).
([skin, disease],	[cat=na, head=Y, mod=Z] $\bowtie$ [X.isa=skin, Y.P=X, Y.isa=disease, Z.P1=Y]).
([disease],	[cat=nc, det=no, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=X] $\bowtie$ [X.isa=disease]).
([skin, disease] [28],	[cat=nc, det=no, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=Y] $\bowtie$ [X.isa=skin, Y.P=X, Y.isa=disease]).
([paper],	[cat=n, det=no, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=X] $\bowtie$ [X.isa=paper]).
([technical, paper],	[cat=n, det=no, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=Y] $\bowtie$ [X.isa=technical, Y.P=X, Y.isa=paper]).
([a, paper],	[cat=n, det=y, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=X] $\bowtie$ [X.det=a, X.isa=paper]).
([jody],	[cat=n, det=y, pers=3, nr=sg, case=na, hum=y, gen=fem, head=X] $\bowtie$ [X.name=jody]).
([he],	[cat=n, det=y, pers=3, nr=sg, case=n, hum=y, gen=male, head=X] $\bowtie$ [X.isa=he]).
([he] [29],	[cat=sbj, dets=y, pers=3, nr=sg, case=n, hum=y, head=X]$\bowtie$[X.isa=he]).

The learned grammar is given below.

Fragment of Learned Grammar, G_{np}
a(H$\bowtie$B-Bo)$--$ >adj(H1$\bowtie$B-Bo),{fi-comp1(H,H1),fi-onto(B-Bo)}.
a(H$\bowtie$B-Bo)$--$ >adv(H1$\bowtie$B-B1),a(H2$\bowtie$B1-Bo),{fi-comp2(H,H1,H2),fi-onto(B-Bo)}.
na(H$\bowtie$B-Bo)$--$ >noun(H1$\bowtie$B-Bo),{fi-comp3(H,H1),fi-onto(B-Bo)}.
na(H$\bowtie$B-Bo)$--$ >na(H1$\bowtie$B-B1),na(H2$\bowtie$B1-Bo),{fi-comp4(H,H1,H2),fi-onto(B-Bo)}.
nc(H$\bowtie$B-Bo)$--$ >noun(H1$\bowtie$B-Bo),{fi-comp5(H,H1),fi-onto(B-Bo)}.
nc(H$\bowtie$B-Bo)$--$ >na(H1$\bowtie$B-B1),nc(H2$\bowtie$B1-Bo),{fi-comp6(H,H1,H2),fi-onto(B-Bo)}.
n(H$\bowtie$B-Bo)$--$ >nc(H1$\bowtie$B-Bo),{fi-comp7(H,H1),fi-onto(B-Bo)}.
n(H$\bowtie$B-Bo)$--$ >a(H1$\bowtie$B-B1),n(H2$\bowtie$B1-Bo),{fi-comp8(H,H1,H2),fi-onto(B-Bo)}.
n(H$\bowtie$B-Bo)$--$ >det(H1$\bowtie$B-B1),n(H2$\bowtie$B1-Bo),{fi-comp9(H,H1,H2),fi-onto(B-Bo)}.
n(H$\bowtie$B-Bo)$--$ >pn(H1$\bowtie$B-Bo),{fi-comp10(H,H1),fi-onto(B-Bo)}.
n(H$\bowtie$B-Bo)$--$ >pro(H1$\bowtie$B-Bo),{fi-comp11(H,H1),fi-onto(B-Bo)}.
sbj(H$\bowtie$B-Bo)$--$ >n(H1$\bowtie$B-Bo),{fi-comp12(H,H1),fi-onto(B-Bo)}.

Sample of learned compositional constraints	
fi-comp4(h, h1, h2) =	{h.cat=na, h.head=h1.mod, h.head=h2.head, h.mod=h2.mod, h1.cat=na, h2.cat=na}
fi-comp7(h, h1) =	{h.cat=n, h.det=no, h.pers=h1.pers, h.nr=h1.nr, h.case=h1.case, h.hum=h1.hum, h.gen=h1.gen, h.count=h1.count, h.head=h1.head, h1.cat=nc, h1.det=no}
fi-comp9(h, h1, h2) =	{h.cat=n, h.det=y, h.pers=h2.pers, h.nr=h2.nr, h.case=h2.case, h.hum=h2.hum, h.gen=h2.gen, h.count=h2.count, h.head=h1.head, h.head=h2.head, h1.cat=det, h2.cat=n, h2.det=no}

28. We allow nondeterminism during learning. The examples *disease* and *skin disease* can be both na and nc.
29. In this experiment, we intentionally did not specify the gen feature in the $subj$ category.

Appendix C. Parsing and Generation

Below we give examples of parsing and generation using the learned grammars and our reversible robust parser.

Parsing using the learned gramamr G_{aux} [30]
input: w = [what, can't, have, been, being]
output: $L_\sigma(w)$[31] = { ([what, can't have, been, being], [cat=av4, int=y, dets=no, case=n, hum=no, aux=be, neg=y, tense=mod, pers=3, nr=_, pf=y, pg=y, headS=X, head=Y]⋈[X.isa=what, Y.mod=can, Y.neg=y, Y.tense=mod, Y.pf=y, Y.pg=y]) }

Generation using the learned gramamr G_{aux}
input: b = [X.isa=what, Y.mod=can, Y.neg=y, Y.tense=mod, Y.pf=y, Y.pg=y]
output: $L_\sigma(b)$= {
([what, can, not, have, been, being], [cat=av4, int=y, dets=no, case=n, hum=no, aux=be, neg=y, tense=mod, pers=3, nr=_, pf=y, pg=y, headS= X, head= Y]⋈[X.isa=what, Y.mod=can, Y.neg=y, Y.tense=mod, Y.pf=y, Y.pg=y]),
([what, can't, have, been, being], [cat=av4, int=y, dets=no, case=n, hum=no, aux=be, neg=y, tense=mod, pers=3, nr=_, pf=y, pg=y, headS= X, head= Y]⋈ [X.isa=what, Y.mod=can, Y.neg=y, Y.tense=mod, Y.pf=y, Y.pg=y]),
([what, cannot, have, been, being], [cat=av4, int=y, dets=no, case=n, hum=no, aux=be, neg=y, tense=mod, pers=3, nr=_, pf=y, pg=y, headS= X, head= Y]⋈ [X.isa=what, Y.mod=can, Y.neg=y, Y.tense=mod, Y.pf=y, Y.pg=y])
}

Parsing using the learned gramamr G_{np} [32]
input: w=[skin, disease, treatment]
output: $L_\sigma(w)$ = {
([skin, disease, treatment], [**cat=n**, det=no, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=Z] ⋈ [X.isa=skin, Y.Py=X, Y.isa=disease, Z.Pz=Y, Z.isa=treatment]),
([skin, disease, treatment], [**cat=nc**, det=no, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=Z] ⋈ [X.isa=skin, Y.Py=X, Y.isa=disease, Z.Pz=Y, Z.isa=treatment]),
~~([skin, disease, treatment], [cat=n, det=no, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=Z] ⋈ [X.isa=skin, Z.Pz1=X, Y.isa=disease, Z.Pz2=Y, Z.isa=treatment])~~[33],
~~([skin, disease, treatment], [cat=nc, det=no, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=Z] ⋈ [X.isa=skin, Z.Pz1=X, Y.isa=disease, Z.Pz2=Y, Z.isa=treatment])~~,
([skin, disease, treatment], [**cat=na**, head=Z, mod=T] ⋈ [X.isa=skin, Y.Py=X, Y.isa=disease, Z.Pz=Y, Z.isa=treatment, T.Pt=Z]),
~~([skin, disease, treatment], [cat=na, head=Z, mod=T] ⋈ [X.isa=skin, Z.Pz1=X, Y.isa=disease, Z.Pz2=Y, Z.isa=treatment, T.Pt=Z])~~
}

30. See Appendix A
31. See definition of $L_\sigma(w)$ and $L_\sigma(b)$ in Section 3.3.3. Here we do not show all the chunks.
32. See Appendix B
33. The cross-out examples means they are rejected by the ontological constraint

Generation using the learned gramamr G_{np}

input: $b1$= [X.isa=skin, Y.Py=X, Y.isa=disease, Z.Pz=Y, Z.isa=treatment]
output: $L_\sigma(b1)$ = {
([skin, disease, treatment], [**cat=n**, det=no, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=Z] $\bowtie$ [X.isa=skin, Y.Py=X, Y.isa=disease, Z.Pz=Y, Z.isa=treatment]),
 	
([skin, disease, treatment], [**cat=nc**, det=no, pers=3, nr=sg, case=na, hum=no, gen=neutr, count=y, head=Z] $\bowtie$ [X.isa=skin, Y.Py=X, Y.isa=disease, Z.Pz=Y, Z.isa=treatment])
}

input: $b2$=[X.isa=skin,Y.Py=X,Y.isa=disease,Z.Pz=Y,Z.isa=treatment,T.Pt=Z]
output: $L_\sigma(b2)$ = {
([skin, disease, treatment], [**cat=na**, head=Z, mod=T] $\bowtie$ [X.isa=skin, Y.Py=X, Y.isa=disease, Z.Pz=Y, Z.isa=treatment, T.Pt=Z])
}

Parsing using the learned gramamr G_{clwh}

input: w = [who, has, been, loving, me]
output: $L_\sigma(w)$ = {
([who, has, been, loving, me], [**cat=wh**, **int=y**, dets=no, hum=y, head=Y, headS=X, headC=Z] $\bowtie$ [X.isa=who, Y.tense=pr, Y.pf=y, Y.pg=y, Y.isa=love, Y.agt=X, Y.pnt=Z, Z.isa=me]) [34],
 	
([who, has, been, loving, me], [**cat=cl**, **int=_**, dets=no, hum=y, head=Y, headS=X, headC=Z] $\bowtie$ [X.isa=who, Y.tense=pr, Y.pf=y, Y.pg=y, Y.isa=love, Y.agt=X, Y.pnt=Z, Z.isa=me])[35]
}

Generation using the learned gramamr G_{clwh}

input: b = [X.isa=who, Y.tense=pr, Y.pf=y, Y.pg=y, Y.isa=love, Y.agt=X, Y.pnt=Z, Z.isa=me]
output: $L_\sigma(b)$ = {
([who, has, been, loving, me], [cat=wh, int=y, dets=no, hum=y, head=Y, headS=X, headC=Z] $\bowtie$ [X.isa=who, Y.tense=pr, Y.pf=y, Y.pg=y, Y.isa=love, Y.agt=X, Y.pnt=Z, Z.isa=me]),
 	
([who, have, been, loving, me], [cat=wh, int=y, dets=no, hum=y, head=Y, headS=X, headC=Z] $\bowtie$ [X.isa=who, Y.tense=pr, Y.pf=y, Y.pg=y, Y.isa=love, Y.agt=X, Y.pnt=Z, Z.isa=me]),
 	
([who, has, been, loving, me], [cat=cl, int=_, dets=no, hum=y, head=Y, headS=X, headC=Z] $\bowtie$ [X.isa=who, Y.tense=pr, Y.pf=y, Y.pg=y, Y.isa=love, Y.agt=X, Y.pnt=Z, Z.isa=me]),
 	
([who, have, been, loving, me], [cat=cl, int=_, dets=no, hum=y, head=Y, headS=X, headC=Z] $\bowtie$ [X.isa=who, Y.tense=pr, Y.pf=y, Y.pg=y, Y.isa=love, Y.agt=X, Y.pnt=Z, Z.isa=me])
}

34. Corresponds to interogative clauses

35. It can be noticed that the attribute "int" is not instantiated. It will become instantiated with the value "no" for relative clauses (i.e., relative clauses are not interrogative clauses). In this experiment we did not use the number attribute in the clause head. This way further agreement between the relative clause and the head noun will not take place (i.e., the rule will overgeneralize).