Grammar Approximation by Representative Sublanguage: A New Model for Language Learning

Smaranda Muresan
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742, USA
smara@umiacs.umd.edu

Owen Rambow
Center for Computational Learning Systems
Columbia University
New York, NY 10027, USA
rambow@cs.columbia.edu

Abstract

We propose a new language learning model that learns a syntactic-semantic grammar from a small number of natural language strings annotated with their semantics, along with basic assumptions about natural language syntax. We show that the search space for grammar induction is a complete grammar lattice, which guarantees the uniqueness of the learned grammar.

1 Introduction

There is considerable interest in learning computational grammars.[1] While much attention has focused on learning syntactic grammars either in a supervised or unsupervised manner, recently there is a growing interest toward learning grammars/parsers that capture semantics as well (Bos et al., 2004; Zettlemoyer and Collins, 2005; Ge and Mooney, 2005).

Learning both syntax and semantics is arguably more difficult than learning syntax alone. In formal grammar learning theory it has been shown that learning from "good examples," or representative examples, is more powerful than learning from all the examples (Freivalds et al., 1993). Haghighi and Klein (2006) show that using a handful of "prototypes" significantly improves over a fully unsupervised PCFG induction model (their prototypes were formed by sequences of POS tags; for example, prototypical NPs were DT NN, JJ NN).

In this paper, we present a new grammar formalism and a new learning method which together address the problem of learning a syntactic-semantic grammar in the presence of a representative sample of strings annotated with their semantics, along with minimal assumptions about syntax (such as syntactic categories). The semantic representation is an ontology-based semantic representation. The annotation of the representative examples does not include the entire derivation, unlike most of the existing syntactic treebanks. The aim of the paper is to present the formal aspects of our grammar induction model.

In Section 2, we present a new grammar formalism, called *Lexicalized Well-Founded Grammars*, a type of constraint-based grammars that combine syntax and semantics. We then turn to the two main results of this paper. In Section 3 we show that our grammars can always be learned from a set of positive representative examples (with no negative examples), and the search space for grammar induction is a complete grammar lattice, which guarantees the uniqueness of the learned grammar. In Section 4, we propose a new computationally efficient model for grammar induction from pairs of utterances and their semantic representations, called *Grammar Approximation by Representative Sublanguage (GARS)*. Section 5 discusses the practical use of our model and Section 6 states our conclusions and future work.

[1]This research was supported by the National Science Foundation under Digital Library Initiative Phase II Grant Number IIS-98-17434 (Judith Klavans and Kathleen McKeown, PIs). We would like to thank Judith Klavans for her contributions over the course of this research, Kathy McKeown for her input, and several anonymous reviewers for very useful feedback on earlier drafts of this paper.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 832–839,
Prague, Czech Republic, June 2007. ©2007 Association for Computational Linguistics

2 Lexicalized Well-Founded Grammars

Lexicalized Well-Founded Grammars (LWFGs) are a type of Definite Clause Grammars (Pereira and Warren, 1980) where: (1) the Context-Free Grammar backbone is extended by introducing a partial ordering relation among nonterminals (well-founded) 2) each string is associated with a syntactic-semantic representation called *semantic molecule*; 3) grammar rules have two types of constraints: one for semantic composition and one for ontology-based semantic interpretation.

The partial ordering among nonterminals allows the ordering of the grammar rules, and thus facilitates the bottom-up induction of these grammars.

The *semantic molecule* is a syntactic-semantic representation of natural language strings $w' = \binom{h}{b}$ where h *(head)* encodes the information required for semantic composition, and b *(body)* is the actual semantic representation of the string. Figure 1 shows examples of semantic molecules for an adjective, a noun and a noun phrase. The representations associated with the lexical items are called *elementary semantic molecules* (I), while the representations built by the combination of others are called *derived semantic molecules* (II). The head of the semantic molecule is a flat feature structure, having at least two attributes encoding the syntactic category of the associated string, cat, and the head of the string, head. The set of attributes is finite and known a priori for each syntactic category. The body of the semantic molecule is a flat, ontology-based semantic representation. It is a logical form, built as a conjunction of atomic predicates $\langle concept \rangle.\langle attr \rangle = \langle concept \rangle$, where variables are either concept or slot identifiers in an ontology. For example, the adjective *major* is represented as $\langle X_1.isa = major, X_2.Y = X_1 \rangle$, which says that the meaning of an adjective is a concept ($X_1.isa = major$), which is a value of a property of another concept ($X_2.Y = X_1$) in the ontology.

The grammar nonterminals are augmented with pairs of strings and their semantic molecules. These pairs are called syntagmas, and are denoted by $\sigma = (w, w') = (w, \binom{h}{b})$. There are 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*. An example of a LWFG rule is given in Figure 1(III). The composition constraints Φ_{comp} applied to the heads of the semantic molecules, form a system of equations that is a simplified version of "path equations" (Shieber et al., 1983), because the heads are flat feature structures. These constraints are learned together with the grammar rules. The ontology-based constraints represent the validation on the ontology, and are applied to the body of the semantic molecule associated with the left-hand side nonterminal. They are not learned. Currently, Φ_{onto} is a predicate which can succeed or fail. When it succeeds, it instantiates the variables of the semantic representation with concepts/slots in the ontology. For example, given the phrase *major damage*, Φ_{onto} succeeds and returns (X_1=MAJOR, X=DAMAGE, Y=DEGREE), while given the phrase *major birth* it fails. We leave the discussion of the ontology constraints for a future paper, since it is not needed for the main result of this paper.

We give below the formal definition of Lexical-

I. Elementary Semantic Molecules

(major/adj)′ = $\begin{pmatrix} \begin{bmatrix} \text{cat} & \text{adj} \\ \text{head} & X_1 \\ \text{mod} & X_2 \end{bmatrix} \\ h_1 \\ b_1 \langle X_1.\text{isa} = \text{major}, X_2.\text{Y}=X_1 \rangle \end{pmatrix}$

(damage/noun)′ = $\begin{pmatrix} \begin{bmatrix} \text{cat} & \text{noun} \\ \text{nr} & \text{sg} \\ \text{head} & X_3 \end{bmatrix} \\ h_3 \\ b_3 \langle X_3.\text{isa} = \text{damage} \rangle \end{pmatrix}$

II. Derived Semantic Molecule

(major damage)′ = $\begin{pmatrix} \begin{bmatrix} \text{cat} & \text{n} \\ \text{nr} & \text{sg} \\ \text{head} & \text{X} \end{bmatrix} \\ h \\ b \langle X_1.\text{isa} = \text{major}, \text{X.Y}=X_1, \text{X.isa}=\text{damage} \rangle \end{pmatrix}$

III. Constraint Grammar Rule

$N(w, \binom{h}{b}) \rightarrow Adj(w_1, \binom{h_1}{b_1}), N(w_2, \binom{h_2}{b_2}) : \Phi_{comp}, \Phi_{onto}(b)$

$\Phi_{comp}(h, h_1, h_2) = \{h.cat = n, h.head = h_1.mod, h.head = h_2.head, h.nr = h_2.nr, h_1.cat = adj, h_2.cat = n\}$

$\Phi_{onto}(b)$ returns X_1=MAJOR, X=DAMAGE, Y=DEGREE from ontology

Figure 1: Examples of two elementary semantic molecules (I), a derived semantic molecule (II) obtained by combining them, and a constraint grammar rule together with the constraints Φ_{comp}, Φ_{onto} (III) .

ized Well-Founded Grammars, except that we do not define formally the constraints due to lack of space (see (Muresan, 2006) for details).

Definition 1. A *Lexicalized Well-Founded Grammar (LWFG)* is a 6-tuple, $G = \langle \Sigma, \Sigma', N_G, \succeq, P_G, S \rangle$, where:

1. Σ is a finite set of terminal symbols.
2. Σ' is a finite set of elementary semantic molecules corresponding to the set of terminal symbols.
3. N_G is a finite set of nonterminal symbols.
4. $\succeq$ is a partial ordering relation among the nonterminals.
5. P_G is a set of constraint rules. A constraint rule is 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 $\sigma = (w, w'), \sigma_i = (w_i, w_i'), 1 \leq i \leq n, w = w_1 \cdots w_n, w' = w_1' \circ \cdots \circ w_n'$, and $\circ$ is the semantic composition operator. For brevity, we denote a rule by $A \rightarrow \beta\colon \Phi$, where $A \in N_G, \beta \in N_G^+$. For the rules whose left-hand side are preterminals, $A(\sigma) \rightarrow$, we use the notation $A \rightarrow \sigma$. There are three types of rules: ordered non-recursive, ordered recursive, and non-ordered rules. A grammar rule $A(\sigma) \rightarrow B_1(\sigma_1), \ldots, B_n(\sigma_n)\colon \Phi(\bar{\sigma})$, is an *ordered rule*, if for all B_i, we have $A \succeq B_i$. In LWFGs, each nonterminal symbol is a left-hand side in at least one ordered non-recursive rule and the empty string cannot be derived from any nonterminal symbol.
6. $S \in N_G$ is the start nonterminal symbol, and $\forall A \in N_G, S \succeq A$ (we use the same notation for the reflexive, transitive closure of $\succeq$).

The relation $\succeq$ is a partial ordering only among nonterminals, and it should not be confused with information ordering derived from the flat feature structures. This relation makes the set of nonterminals well-founded, which allows the ordering of the grammar rules, as well as the ordering of the syntagmas generated by LWFGs.

Definition 2. Given a LWFG, G, the *ground syntagma derivation* relation, $\stackrel{*G}{\Rightarrow}$,[2] is defined as: $\frac{A \rightarrow \sigma}{A \stackrel{*G}{\Rightarrow} \sigma}$ (if $\sigma = (w, w'), w \in \Sigma, w' \in \Sigma'$, i.e., A is a preterminal), and $\frac{B_i \stackrel{*G}{\Rightarrow} \sigma_i,\ i=1,\ldots,n, \quad A(\sigma) \rightarrow B_1(\sigma_1), \ldots, B_n(\sigma_n)\colon \Phi(\bar{\sigma})}{A \stackrel{*G}{\Rightarrow} \sigma}$.

In LWFGs all syntagmas $\sigma = (w, w')$, derived from a nonterminal A have the same category of their semantic molecules w'.[3]

The language of a grammar G is the set of all syntagmas generated from the start symbol S, i.e., $L(G) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, S \stackrel{*G}{\Rightarrow} \sigma\}$. The *set of all syntagmas* generated by *a grammar* G is $L_\sigma(G) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, \exists A \in N_G, A \stackrel{*G}{\Rightarrow} \sigma\}$. Given a LWFG G we call a set $E_\sigma \subseteq L_\sigma(G)$ a sublanguage of G. Extending the notation, given a LWFG G, the set of syntagmas generated by a *rule* $(A \rightarrow \beta\colon \Phi) \in P_G$ is $L_\sigma(A \rightarrow \beta\colon \Phi) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, (A \rightarrow \beta\colon \Phi) \stackrel{*G}{\Rightarrow} \sigma\}$, where $(A \rightarrow \beta\colon \Phi) \stackrel{*G}{\Rightarrow} \sigma$ denotes the ground derivation $A \stackrel{*G}{\Rightarrow} \sigma$ obtained using the rule $A \rightarrow \beta\colon \Phi$ in the last derivation step (we have bottom-up derivation). We will use the short notation $L_\sigma(r)$, where r is a grammar rule.

Given a LWFG G and a sublanguage E_σ (not necessarily of G) we denote by $\mathbb{S}(G) = L_\sigma(G) \cap E_\sigma$, the set of syntagmas generated by 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 set of syntagmas generated by r *reduced to the sublanguage* E_σ.

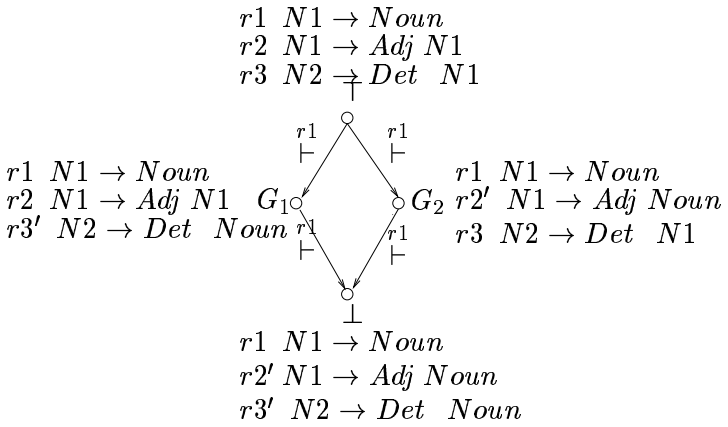
As we have previously mentioned, the partial ordering among grammar nonterminals allows the ordering of the syntagmas generated by the grammar, which allows us to define the *representative examples* of a LWFG.

Representative Examples. Informally, the representative examples E_R of a LWFG, G, are the simplest syntagmas ground-derived by the grammar G, i.e., for each grammar rule there exist a syntagma which is ground-derived from it in the minimum number of steps. Thus, the size of the representative example set is equal with the size of the set of grammar rules, $|E_R| = |P_G|$.

This set of representative examples is used by the grammar learning model to generate the candidate hypotheses. For generalization, a larger sublanguage $E_\sigma \supseteq E_R$ is used, which we call representative sublanguage.

[2] The ground derivation ("reduction" in (Wintner, 1999)) can be viewed as the bottom-up counterpart of the usual derivation.

[3] This property is used for determining the lhs nonterminal of the learned rule.



Σ = {the, noise, loud, clear}
E_R={*noise, loud noise, the noise*}
E_σ=E_R ∪ {*clear loud noise, the loud noise*}
$\mathbb{S}(\top)$= E_σ
$\mathbb{S}(G1)$= E_R ∪ {*clear loud noise*}
$\mathbb{S}(G2)$= E_R ∪ {*the loud noise*}
$\mathbb{S}(\bot)$= E_R

Rule specialization steps

$r_2 \overset{r_1}{\vdash} r_2'$

$r_3 \overset{r_1}{\vdash} r_3'$

Rule generalization steps

$r_2' \overset{r_1}{\dashv} r_2$

$r_3' \overset{r_1}{\dashv} r_3$

Figure 2: Example of a simple grammar lattice. All grammars generate E_R, and only $\top$ generates E_σ (Σ is a common lexicon for all the grammars)

3 A Grammar Lattice as a Search Space for Grammar Induction

In this section we present a class of Lexicalized Well-Founded Grammars that form a complete lattice. This grammar lattice is the search space for our grammar induction model, which we present in Section 4. An example of a grammar lattice is given in Figure 2, where for simplicity, we only show the context-free backbone of the grammar rules, and only strings, not syntagmas. Intuitively, the grammars found lower in the lattice are more specialized than the ones higher in the lattice. For learning, E_R is used to generate the most specific hypotheses (grammar rules), and thus all the grammars should be able to generate those examples. The sublanguage E_σ is used during generalization, thus only the most general grammar, $\top$, is able to generate the entire sublanguage. In other words, the generalization process is bounded by E_σ, that is why our model is called Grammar Approximation by Representative Sublanguage.

There are two properties that LWFGs should have in order to form a complete lattice: 1) they should be unambiguous, and 2) they should preserve the parsing of the representative example set, E_R. We define these two properties in turn.

Definition 3. A LWFG, G, is *unambiguous* w.r.t. a sublanguage $E_\sigma \subseteq L_\sigma(G)$ if $\forall \sigma \in E_\sigma$ there is one and only one rule that derives σ.

Since the unambiguity is relative to a set of syntagmas (pairs of strings and their semantic molecules) and not to a set of natural language strings, the requirement is compatible with modeling natural language. For example, an ambiguous string such as *John saw the man with the telescope* corresponds to two unambiguous syntagmas.

In order to define the second property, we need to define *the rule specialization step* and *the rule generalization step* of unambiguous LWFGs, such that they are *E_R-parsing-preserving* and are the inverse of each other. The property of E_R-parsing-preserving means that both the initial and the specialized/generalized rules ground-derive the same syntagma, $\sigma_A \in E_R$.

Definition 4. The *rule specialization step*:

$$\frac{A(\sigma_A)\rightarrow\alpha B(\sigma_B^*)\gamma\colon \Phi_A \qquad B(\sigma_B)\rightarrow\beta\colon \Phi_B}{A(\sigma_A)\rightarrow\alpha\beta\gamma\colon \Phi'_A}$$

is *E_R-parsing-preserving*, if there exists $\sigma_A \in E_R$ and $r_{gen} \overset{*G}{\Rightarrow} \sigma_A$ and $r_{spec} \overset{*G'}{\Rightarrow} \sigma_A$, where $r_{gen} = A(\sigma_A) \rightarrow \alpha B(\sigma_B^*)\gamma\colon \Phi_A$, $r_B = B(\sigma_B) \rightarrow \beta\colon \Phi_B$, and $r_{spec} = A(\sigma_A) \rightarrow \alpha\beta\gamma\colon \Phi'_A$. We write $r_{gen} \overset{r_B}{\vdash} r_{spec}$.

The *rule generalization step* :

$$\frac{A(\sigma_A)\rightarrow\alpha\beta\gamma\colon \Phi'_A \qquad B(\sigma_B)\rightarrow\beta\colon \Phi_B}{A(\sigma_A)\rightarrow\alpha B(\sigma_B^*)\gamma\colon \Phi_A}$$

is *E_R-parsing-preserving*, if there exists $\sigma_A \in E_R$ and $r_{spec} \overset{*G'}{\Rightarrow} \sigma_A$ and $r_{gen} \overset{*G}{\Rightarrow} \sigma_A$. We write $r_{spec} \overset{r_B}{\dashv} r_{gen}$.

Since σ_A is a representative example, it is derived in the minimum number of derivation steps, and thus the rule r_B is always an ordered, non-recursive rule.

The goal of the rule specialization step is to obtain a new target grammar G' from G by modifying a rule of G. Similarly, the goal of the rule generalization step is to obtain a new target grammar G from G' by modifying a rule of G'. They are not to be taken as the derivation/reduction concepts in parsing. The specialization/generalization steps are the inverse of each other. From both the specialization and the generalization step we have that: $L_\sigma(r_{gen}) \supseteq L_\sigma(r_{spec})$.

In Figure 2, the specialization step $r_2 \overset{r_1}{\vdash} r_2'$ is E_R-parsing-preserving, because the rule r_2' ground-derives the syntagma *loud noise*. If instead we would have a specialization step $r_2 \overset{r_2}{\vdash} r_2''$ ($r_2'' = N1 \rightarrow Adj\ Adj\ N1$), it would not be E_R-parsing-preserving since the syntagma *loud noise* could no longer be ground-derived from the rule r_2'' (which requires two adjectives).

Definition 5. A grammar G' is *one-step specialized* from a grammar G, $G \overset{r_1}{\vdash} G'$, if $\exists r, r_1 \in P_G$ and $\exists r', r_1 \in P_{G'}$, s.t. $r \overset{r_1}{\vdash} r'$, and $\forall q \neq r, q \in P_G$ iff $q \in P_{G'}$. A grammar G' is *specialized* from a grammar G, $G \overset{*}{\vdash} G'$, if it is obtained from G in n-specialization steps: $G \overset{r_1}{\vdash} \cdots \overset{r_n}{\vdash} G'$, where n is finite. We extend the notation so that we have $G \overset{*}{\vdash} G$. Similarly, we define the concept of a grammar G *g*eneralized from a grammar G', $G' \overset{*}{\dashv} G$ using the rule generalization step.

In Figure 2, the grammar $\perp$ is one-step specialized from the grammar G_1, i.e., $G_1 \overset{r_1}{\vdash} \perp$, since $\perp$ preserve the parsing of the representative examples E_R. A grammar which contains the rule $r_2'' = N1 \rightarrow Adj\ Adj N1$ instead of r_2' is not specialized from the grammar G_1 since it does not preserve the parsing of the representative example set, E_R. Such grammars will not be in the lattice.

In order to define the grammar lattice we need to introduce one more concept: a *normalized* grammar w.r.t. a sublanguage.

Definition 6. A LWFG G is called *normalized* w.r.t. a sublanguage E_σ (not necessarily of G), if none of the grammar rules r_{spec} of G can be further generalized to a rule r_{gen} by the rule generalization step such that $\mathbb{S}(r_{spec}) \subset \mathbb{S}(r_{gen})$.

In Figure 2, grammar $\top$ is normalized w.r.t. E_σ, while $\perp$, G_1 and G_2 are not.

We now define a grammar lattice $\mathfrak{L}$ which will be the search space for our grammar learning model. We first define the set of lattice elements $\mathcal{L}$.

Let $\top$ be a LWFG, normalized and unambiguous w.r.t. a sublanguage $E_\sigma \subseteq L_\sigma(\top)$ which includes the representative example set E_R of the grammar $\top$ ($E_\sigma \supseteq E_R$). Let $\mathcal{L} = \{G | \top \overset{*}{\vdash} G\}$ be the set of grammars specialized from $\top$. We call $\top$ the *top element* of $\mathcal{L}$, and $\perp$ the *bottom element* of $\mathcal{L}$, if $\forall G \in \mathcal{L}, \top \overset{*}{\vdash} G \wedge G \overset{*}{\vdash} \perp$. The bottom element, $\perp$, is the grammar specialized from $\top$, such that the right-hand side of all grammar rules contains only preterminals. We have $\mathbb{S}(\top) = E_\sigma$ and $\mathbb{S}(\perp) \supseteq E_R$.

The grammars in $\mathcal{L}$ have the following two properties (Muresan, 2006):

- For two grammars G and G', we have that G' is specialized from G if and only if G is generalized from G', with $L_\sigma(G) \supseteq L_\sigma(G')$.
- All grammars in $\mathcal{L}$ preserve the parsing of the representative example set E_R.

Note that we have that for $G, G' \in \mathcal{L}$, if $G \overset{*}{\vdash} G'$ then $\mathbb{S}(G) \supseteq \mathbb{S}(G')$.

The system $\mathfrak{L} = \langle \mathcal{L}, \overset{*}{\vdash} \rangle$ is a complete grammar lattice (see (Muresan, 2006) for the full formal proof). In Figure 2 the grammars G_1, G_2, $\top$, $\perp$ preserve the parsing of the representative examples E_R. We have that $\top \overset{r_1}{\vdash} G_1$, $\top \overset{r_1}{\vdash} G_2$, $G_2 \overset{r_1}{\vdash} \perp$, $G_1 \overset{r_1}{\vdash} \perp$ and $\top \overset{*}{\vdash} \perp$. Due to space limitation we do not define here the *least upper bound* (lub), $\curlyvee$ and the *greatest lower bound* (glb), $\curlywedge$ operators, but in this example $\top = G_1 \curlyvee G_2$, $\perp = G_1 \curlywedge G_2$.

In oder to give a learnability theorem we need to show that $\perp$ and $\top$ elements of the lattice can be built. First, an assumption in our learning model is that the rules corresponding to the grammar preterminals are given. Thus, for a given set of representative examples, E_R, we can build the grammar $\perp$ using a bottom-up robust parser, which returns partial analyses (chunks) if it cannot return a full parse. In order to soundly build the $\top$ element of the grammar lattice from the $\perp$ grammar through generalization, we must give the definition of a grammar G *conformal* w.r.t. E_σ.

Definition 7. A LWFG G is *conformal* w.r.t. a sublanguage $E_\sigma \subseteq L_\sigma(G)$ iff G is normalized and unambiguous w.r.t. E_σ and the rule specialization step guarantees that $\mathbb{S}(r_{gen}) \supset \mathbb{S}(r_{spec})$ for all grammars specialized from G.

The only rule generalization steps allowed in the grammar induction process are those which guarantee the same relation $\mathbb{S}(r_{spec}) \subset \mathbb{S}(r_{gen})$, which ensures that all the generalized grammars belong to the grammar lattice.

In Figure 2, $\top$ is conformal to the given sublanguage E_σ. If the sublanguage were $E^*_\sigma = E_R \cup \{$*clear loud noise*$\}$ then $\top$ would not be conformal to E^*_σ since $\mathbb{S}(\top) = \mathbb{S}(G_1) = E^*_\sigma$ and thus the specialization step would not satisfy the relation $\mathbb{S}(N2 \rightarrow Det\ N1) \supset \mathbb{S}(N2 \rightarrow Det\ Noun)$. During learning, the generalization step cannot generalize from grammar G_1 to $\top$.

Theorem 1 (Learnability Theorem). *If E_R is the set of representative examples associated with a LWFG G conformal w.r.t. a sublanguage $E_\sigma \supseteq E_R$, then G can always be learned from E_R and E_σ as the grammar lattice top element ($\top = G$).*

The proof is given in (Muresan, 2006).

If the hypothesis of Theorem 1 holds, then any grammar induction algorithm that uses the complete lattice search space can converge to the lattice top element, using different search strategies. In the next section we present our new model of grammar learning which relies on the property of the search space as grammar lattice.

4 Grammar Induction Model

Based on the theoretical foundation of the hypothesis search space for LWFG learning given in the previous section, we define our grammar induction model. First, we present the LWFG induction as an Inductive Logic Programming problem. Second, we present our new relational learning model for LWFG induction, called *Grammar Approximation by Representative Sublanguage (GARS)*.

4.1 Grammar Induction Problem in ILP-setting

Inductive Logic Programming (ILP) is a class of relational learning methods concerned with inducing first-order Horn clauses from examples and background knowledge. Kietz and Džeroski (1994) have formally defined the *ILP-learning problem* as the tuple $\langle \vdash, LB, LE, LH \rangle$, where $\vdash$ is the provability relation (also called the generalization model), LB is the language of the background knowledge, LE is the language of the (positive and negative) examples, and LH is the hypothesis language. The general ILP-learning problem is undecidable. Possible choices to restrict the ILP-problem are: the provability relation, $\vdash$, the background knowledge and the hypothesis language. Research in ILP has presented positive results only for very limited subclasses of first-order logic (Kietz and Džeroski, 1994; Cohen, 1995), which are not appropriate to model natural language grammars.

Our grammar induction problem can be formulated as an ILP-learning problem $\langle \vdash, LB, LE, LH \rangle$ as follows:

- The provability relation, $\vdash$, is given by robust parsing, and we denote it by $\vdash_{rp}$. We use the "parsing as deduction" technique (Shieber et al., 1995). For all syntagmas we can say in polynomial time whether they belong or not to the grammar language. Thus, using the $\vdash_{rp}$ as generalization model, our grammar induction problem is decidable.

- The language of background knowledge, LB, is the set of LWFG rules that are already learned together with elementary syntagmas (i.e., corresponding to the lexicon), which are ground atoms (the variables are made constants).

- The language of examples, LE are syntagmas of the representative sublanguage, which are ground atoms. We only have positive examples.

- The hypothesis language, LH, is a LWFG lattice whose top element is a conformal grammar, and which preserve the parsing of representative examples.

4.2 Grammar Approximation by Representative Sublanguage Model

We have formulated the grammar induction problem in the ILP-setting. The theoretical learning model,

called *Grammar Approximation by Representative Sublanguage (GARS)*, can be formulated as follows:
Given:

- a representative example set E_R, lexically consistent (i.e., it allows the construction of the grammar lattice $\perp$ element)
- a finite sublanguage E_σ, conformal and thus unambiguous, which includes the representative example set, $E_\sigma \supseteq E_R$. We called this sublanguage, the *representative sublanguage*

Learn a grammar G, using the above ILP-learning setting, such that G is unique and $E_\sigma \subseteq L_\sigma(G)$.

The hypothesis space is a complete grammar lattice, and thus the uniqueness property of the learned grammar is guaranteed by the learnability theorem (i.e., the learned grammar is the lattice top element). This learnability result extends significantly the class of problems learnable by ILP methods.

The GARS model uses two polynomial algorithms for LWFG learning. In the first algorithm, the learner is presented with an ordered set of representative examples (syntagmas), i.e., the examples are ordered from the simplest to the most complex. The reader should remember that for a LWFG G, there exists a partial ordering among the grammar nonterminals, which allows a total ordering of the representative examples of the grammar G. Thus, in this algorithm, the learner has access to the ordered representative syntagmas when learning the grammar. However, in practice it might be difficult to provide the learner with the "true" order of examples, especially when modeling complex language phenomena. The second algorithm is an iterative algorithm that learns starting from a random order of the representative example set. Due to the property of the search space, both algorithms converge to the same target grammar.

Using ILP and theory revision terminology (Greiner, 1999), we can establish the following analogy: syntagmas (examples) are "labeled queries", the LWFG lattice is the "space of theories", and a LWFG in the lattice is "a theory." The first algorithm learns from an "empty theory", while the second algorithm is an instance of "theory revision", since the grammar ("theory") learned during the first iteration, is then revised, by deleting and adding rules.

Both of these algorithms are cover set algorithms. In the first step the most specific grammar rule is generated from the current representative example. The category name annotated in the representative example gives the name of the lhs nonterminal (predicate invention in ILP terminology), while the robust parser returns the minimum number of chunks that cover the representative example. In the second step this most specific rule is generalized using as performance criterion the number of the examples in E_σ that can be parsed using the candidate grammar rule (hypothesis) together with the previous learned rules. For the full details for these two algorithms, and the proof of their polynomial efficiency, we refer the reader to (Muresan, 2006).

5 Discussion

A practical advantage of our GARS model is that instead of writing syntactic-semantic grammars by hand (both rules and constraints), we construct just a small annotated treebank - utterances and their semantic molecules. If the grammar needs to be refined, or enhanced, we only refine, or enhance the representative examples/sublanguage, and not the grammar rules and constraints, which would be a more difficult task.

We have built a framework to test whether our GARS model can learn diverse and complex linguistic phenomena. We have primarily analyzed a set of definitional-type sentences in the medical domain. The phenomena covered by our learned grammar includes complex noun phrases (including noun compounds, nominalization), prepositional phrases, relative clauses and reduced relative clauses, finite and non-finite verbal constructions (including, tense, aspect, negation, and subject-verb agreement), copula *to be*, and raising and control constructions. We also learned rules for wh-questions (including long-distance dependencies). In Figure 3 we show the ontology-level representation of a definition-type sentence obtained using our learned grammar. It includes the treatment of reduced relative clauses, raising construction (*tends to persist*, where *virus* is not the argument of *tends* but the argument of *persist*), and noun compounds. The learned grammar together with a semantic interpreter targeted to terminological knowledge has been used in an acquisition-query experiment, where the answers are at the concept level (the querying is a graph

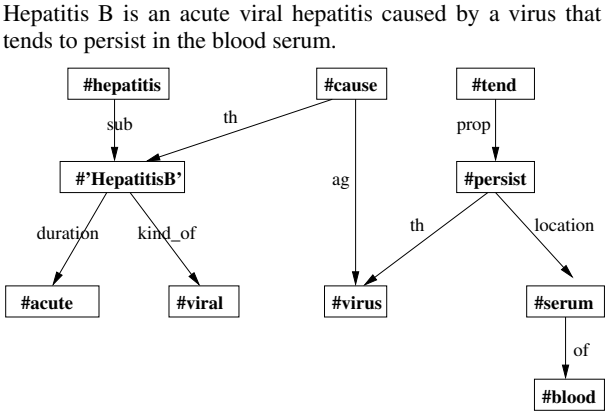


Figure 3: A definition-type sentence and its ontology-based representation obtained using our learned LWFG

matching problem where the "wh-word" matches the answer concept). A detailed discussion of the linguistic phenomena covered by our learned grammar using the GARS model, as well as the use of this grammar for terminological knowledge acquisition, is given in (Muresan, 2006).

To learn the grammar used in these experiments we annotated 151 representative examples and 448 examples used as a representative sublanguage for generalization. Annotating these examples requires knowledge about categories and their attributes. We used 31 categories (nonterminals) and 37 attributes (e.g., category, head, number, person). In this experiment, we chose the representative examples guided by the type of phenomena we wanted to modeled and which occurred in our corpus. We also used 13 lexical categories (i.e., parts of speech). The learned grammar contains 151 rules and 151 constraints.

6 Conclusion

We have presented Lexicalized Well-Founded Grammars, a type of constraint-based grammars for natural language specifically designed to enable learning from representative examples annotated with semantics. We have presented a new grammar learning model and showed that the search space is a complete grammar lattice that guarantees the uniqueness of the learned grammar. Starting from these fundamental theoretical results, there are several directions into which to take this research. A first obvious extension is to have probabilistic-LWFGs. For example, the ontology constraints might not be "hard" constraints, but "soft" ones (because language expressions are more or less likely to be used in a certain context). Investigating where to add probabilities (ontology, grammar rules, or both) is part of our planned future work. Another future extension of this work is to investigate how to automatically select the representative examples from an existing treebank.

References

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING-04*.

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

Rusins Freivalds, Efim B. Kinber, and Rolf Wiehagen. 1993. On the power of inductive inference from good examples. *Theoretical Computer Science*, 110(1):131–144.

R. Ge and R.J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of CoNLL-2005*.

Russell Greiner. 1999. The complexity of theory revision. *Artificial Intelligence Journal*, 107(2):175–217.

Aria Haghighi and Dan Klein. 2006. Prototype-driven grammar induction. In *Proceedings of ACL'06*.

Jörg-Uwe Kietz and Sašo Džeroski. 1994. Inductive logic programming and learnability. *ACM SIGART Bulletin.*, 5(1):22–32.

Smaranda Muresan. 2006. *Learning Constraint-based Grammars from Representative Examples: Theory and Applications*. Ph.D. thesis, Columbia University. http://www1.cs.columbia.edu/∼smara/muresan_thesis.pdf.

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

Stuart Shieber, Hans Uszkoreit, Fernando Pereira, Jane Robinson, and Mabry Tyson. 1983. The formalism and implementation of PATR-II. In Barbara J. Grosz and Mark Stickel, editors, *Research on Interactive Acquisition and Use of Knowledge*, pages 39–79. SRI International, Menlo Park, CA, November.

Stuart Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1-2):3–36.

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

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI-05*.