Learning Constraint-based Grammars from Representative Examples: Theory and Applications

Smaranda Muresan

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2006

Abstract

Learning Constraint-based Grammars from Representative Examples: Theory and Applications

Smaranda Muresan

Computationally efficient models for natural language understanding can have a wide variety of applications starting from text mining and question answering, to natural language interfaces to databases. Constraint-based grammar formalisms have been widely used for deep language understanding. Yet, one serious obstacle for their use in real world applications is that these formalisms have overlooked an important requirement: learnability. Currently, there is a poor match between these grammar formalisms and existing learning methods.

This dissertation defines a new type of constraint-based grammars, Lexicalized Well-Founded Grammars (LWFGs), which allow deep language understanding and are learnable. These grammars model both syntax and semantics and have constraints at the rule level for semantic composition and semantic interpretation. The interpretation constraints allow access to meaning during language processing. They establish links between linguistic expressions and the entities they refer to in the real world. We use an ontology-based interpretation, proposing a semantic

representation that can be conceived as an ontology query language. This representation is sufficiently expressive to represent many aspects of language and yet sufficiently restrictive to support learning and tractable inferences.

In this thesis, we propose a new relational learning model for LWFG induction. The learner is presented with a small set of positive representative examples, which consist of utterances paired with their semantic representations. We have proved that the search space for grammar induction is a complete grammar lattice, which allows the construction and generalization of the hypotheses and guarantees the uniqueness of the solution, regardless of the order of learning. We have proved a learnability theorem and have provided polynomial algorithms for LWFG induction, proving their soundness. The learnability theorem extends significantly the class of problems learnable by Inductive Logic Programming methods.

In this dissertation, we have implemented a system that represents an experimental platform for all the theoretical algorithms. The system has the practical advantage of implementing sound grammar revision and grammar merging, which allow an incremental coverage of natural language fragments. We have provided qualitative evaluations that cover the following issues: coverage of diverse and complex linguistic phenomena; terminological knowledge acquisition from natural language definitions; and handling of both precise and vague questions with precise answers at the concept level.

Contents

List of Figures

List of Tables

Acknowledgments

This thesis would not have been possible without the guidance, support and love of several people. I am particularly grateful to my dissertation advisors Judith Klavans and Owen Rambow, for providing invaluable direction and support throughout my thesis work. I thank Judith for believing in my topic at the time when it was just an attempt, for encouraging and guiding me along the way. I thank Owen for being the perceptive and very caring advisor of the latest stages of my work, for giving me precious and detailed advice.

I am indebted to my thesis committee: Graeme Hirst, Julia Hirschberg, Judith Klavans, Kathleen McKeown, and Owen Rambow. Their questions, suggestions and ideas will be foremost in my mind as this project evolves. In particular, I want to thank Kathy and Julia for their constant advice throughout these years, which profoundly influenced my research and my academic life. I want to thank Graeme for his detailed feedback and constructive criticisms, which helped me shape the presentation of this dissertation.

I would like to express my gratitude to my dear friends Regina Barzilay, Pablo Duboue, Noemie Elhadad, Michel Galley, Carl Sable and Victoria Tzotzkova for the intellectual atmosphere, unfailing help, and encouragement that have been with me throughout this thesis work. They all made my graduate life much happier.

I would like to thank my other friends and colleagues in the Columbia NLP group with whom I spent these beautiful years of my life: Min-Yen Kan, Dave Evans, Elena Filatova, David Elson, James Shaw, Simone Teufel, Sasha Blair-Goldensohn, Rebecca Passonneau, Melania Degeratu, Ani Nenkova, Barry Schiffman, Shimei Pan, Hongyan Jing, Vassielios Hatzivassiloglou, Brian Whit-

man, Dragomir Radev, Nina Wacholder, Melissa Holcombe, Greg Whalen, Augustin Gravano, Sameer Maskey, Aaron Harnly, Jackson Liscombe, Frank Enos, Andrew Rosenberg, Sergey Sigelman, Gabriel Illouz, Hong Yu, Nizar Habash, Martin Jansche, Mona Diab, Lokesh Shrestha, Fadi Biadsy, John Chen, Peter Davis.

To my friends in the CS department: Amelie Marian, Panagiotis Ipeirotis, Eugene Agichtein, Kathy Dunn, Alpa Shah, Liz Chen, Sinem Guven, and Alejandro Troccoli many thanks for their friendship and support.

The entire faculty of the Computer Science Department deserve thanks for their moral and intellectual support over the years, with special appreciation to Luis Gravano and Chris Okasaki. I am very grateful to the administrative staff: Rosemary Addarich, Alice Cueba, Twinkle Edwards, Patricia Hervey, Sandra Morris, Remiko Moss, Anamika Singh and Mary van Starrex whose help was invaluable. I would like to acknowledge the financial support of NSF IIS-9817434 (DLI2).

I could not have done this work without the unconditional love and support of my family. This thesis is dedicated to them. To my mother who has always encouraged me and helped me get through the toughest days with a smile. To my father who gave me the love for research and academic life, and who continued to fuel this love through my graduate years. To my sister who constantly inspired my work on language, and who never failed to be there for me.

To my parents and my sister,

Chapter 1

Introduction

The question "What does it mean to learn language?" is one of the greatest topics of scientific inquiry. The problem has attracted many researchers in linguistics, computer science, and cognitive science, and attempts to answer this question vary greatly from discipline to discipline.

In this dissertation, we consider language learning as the problem of learning grammars which capture syntax and semantics and which allow access to meaning during language processing (i.e., parsing and generation). In addition, we are interested to investigate whether computers can learn such grammars for deep language understanding. For this, we need to provide a computationally efficient learning model.

In order to *build* such a model for language learning, we need to address the following questions:

- What representation is appropriate to provide a deep semantic analysis of natural language and to allow learning and tractable inferences at the same

time

- What properties should the grammar have to include semantics and to be learnable?

- What learning paradigm is needed?

In order to *evaluate* the utility of the learning model, we need to address the following question: What applications does our language learning model have? Figure 1.1 presents some scenarios that can be used to answer this question. The immediate application that comes to mind is a grammar development framework. Linguists are usually concerned with particular linguistic phenomena, such as raising and control, long-distance dependencies, or the predicative and equative meaning of the copula *to be* (Figure 1.1(a)). The second possible application is the acquisition of knowledge from text. In Figure 1.1(b) we present the task of acquiring terminological knowledge from a set of definitions in a particular domain (i.e., medical domain). The third possible application is a question-answering type of task, where both precise and vague questions can be asked w.r.t. an utterance or a set of utterances, and precise answers at the concept level are required (Figure 1.1(c)).

As can be seen from Figure 1.1, the issues in all these scenarios are interleaved: interpretation of copula *to be* appears in definitions, long-distance dependencies might appear in questions, and real-world applications need ways to deal with various linguistic phenomena that might be particular for a domain or task. Thus, our model should be expressive enough and extendible so that different phenomena can be investigated and modeled. Another important aspect is grammar

RAISING/CONTROL CONSTRUCTIONS
Skeptics continue to question your hypothesis.
Reporters tried to interview the candidate.

LONG-DISTANCE DEPENDENCIES
What does the president seem to try to get _ from the farmers?

COPULA "To be"

A. Predicative	**B. Equative**
Pat is smart.	Pat is the mayor.
Pat is a graduate student.	

(a) Linguistic Analysis

Hepatitis is a disease caused by infectious or toxic agents and characterized by jaundice, fever and liver enlargement.

Hepatitis A is an acute but benign viral hepatitis caused by a virus that does not persist in the blood serum.

Hepatitis B is an acute viral hepatitis caused by a virus that tends to persist in the blood serum.

$\Longrightarrow$ *Terminological Knowledge*

(b) Terminological knowledge acquisition from definitions

Q1: What is characterized by jaundice and liver enlargement?
A1: hepatitis

Q2: What is caused by a virus that persists in the blood serum?
A2: Hepatitis B

Q3: What is caused by **something** that does not persist in the blood serum?
A3: Hepatitis A (something=virus)

Monsanto's president who seems to try to get royalties from the farmers who grew genetically modified soya illegally will be giving a briefing for the Brazilian media tomorrow at noon .

Q: What does the president seem to try to get from the farmers?
A: royalties

(c) Precise/vague questions and precise answers at the concept level

Figure 1.1: Scenarios for evaluating the utility of the learning model

merging: how can different grammars be merged in a sound way? This is useful when different aspects of the grammar are developed/learned separately.

In the next section we present the main assumptions we made in building the grammar learning model. In Section 1.2 we give an overview of the research solutions for answering the questions posed in this section. Section 1.3 describes the contributions of this thesis and Section 1.4 is a guide to the remainder of this dissertation.

1.1 Basic Assumptions

Throughout this dissertation we make several basic assumptions.

Ontology as repository of meanings. In order to provide access to meaning during parsing, we assume that the meaning is encoded by an ontology, which will be connected to the grammar through a set of constraints. The focus on ontology is motivated by the need to interpret language in terms of the language-independent concepts of some underlying domain of discourse. Using an ontology to represent meaning has been widely considered in the AI community and in computational linguistics as well (Bateman, 1992; Gruber, 1993; Chaudhri et al., 1998; McDonald, 2001).

A Priori Categories. The second assumption made in this dissertation is that the learner has knowledge about utterance categories. We use both categories that represent the form of the constituents (e.g., Noun, Verb, Noun Phrases, Verb Phrase, etc.) and functional categories of constituents (e.g., Subject, Object, Adjunct) together with *a priori* knowledge about agreement among categories. How-

ever, establishing the exact nature of what categories are needed is outside the scope of this dissertation, and we hope that the learning system can be used by linguists to experiment with different types of categories.

Robust parser as innate inference engine. We assume that the learner has access to a robust, active chart parser (Kay, 1973), during the grammar induction. At the initial state, the background knowledge contains the rules corresponding to preterminals (i.e., these are not learned). The parser is used both during the generation of candidate hypotheses (grammar rules), and during their evaluation in order to choose the best performing one. The robust parser provides all the chunks of an utterance needed to build the candidate hypotheses. We will return to the role of the parser when we talk about our learning paradigm, in the next section.

No noisy data for learning. In this dissertation, we use the "no noisy data" assumption, since the learning is done from a small number of positive annotated examples. This is similar to the assumptions made by other learning frameworks, such as the initial PAC-learning model (Valiant, 1984) and Gold model (Gold, 1967).

1.2 Overview of the Research Results

In this dissertation, we propose a new computationally efficient model for grammar learning, called *Grammar Approximation by Representative Sublanguage (GARS)*. In this model, the language is taken to be a set of strings together with their syntactico-semantic representations. The learner is presented with a set of positive *representative examples* of the target language, and a set of positive examples used

for generalization, which we refer to as a *representative sublanguage*. The representative sublanguage is conformal with respect to the grammar and includes the representative examples. The task of the learner is to induce a grammar that generates the target language. This dissertation defines a new type of constraint-based grammars, *Lexicalized Well-Founded Grammars (LWFGs)*, which allow deep language understanding and are always learnable under the GARS model, that is, the learning always converges to the target grammar. We show that the search space is a grammar lattice and we provide polynomial algorithms for grammar induction and prove they are correct.

In the remainder of this section we describe our new syntactic-semantic representation, our grammar formalism, and our learning paradigm and sketch the utility of the learning model.

1.2.1 Representation of Natural Language

This section summarizes the answers to the question "What representation is appropriate to provide a deep semantic analysis of natural language and to allow learning and tractable inferences at the same time?"

In this dissertation, we introduce a new representation for natural language strings, *semantic molecule*. The semantic molecule is a type of feature structure, $\binom{h}{b}$, where: h *(head)* encodes the information required for composition, and b *(body)* is the actual semantic representation of the string.

Figure 1.2 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

I. Elementary Semantic Molecules

$$(\text{major/adj})' = \left(\begin{array}{l} \begin{bmatrix} \mathsf{cat} & \mathsf{adj} \\ \mathsf{head} & X_1 \\ \mathsf{mod} & X_2 \end{bmatrix}_{h_1} \\ \left\langle X_1.\mathsf{isa} = \mathsf{major},\ X_2.\mathsf{Y}{=}X_1 \right\rangle_{b_1} \end{array} \right)$$

$$(\text{damage/noun})' = \left(\begin{array}{l} \begin{bmatrix} \mathsf{cat} & \mathsf{noun} \\ \mathsf{nr} & \mathsf{sg} \\ \mathsf{head} & X_3 \end{bmatrix}_{h_3} \\ \left\langle X_3.\mathsf{isa} = \mathsf{damage} \right\rangle_{b_3} \end{array} \right)$$

II. Derived Semantic Molecule

$$(\text{major damage})' = \left(\begin{array}{l} \begin{bmatrix} \mathsf{cat} & \mathsf{n} \\ \mathsf{nr} & \mathsf{sg} \\ \mathsf{head} & \mathsf{X} \end{bmatrix}_{h} \\ \left\langle X_1.\mathsf{isa} = \mathsf{major},\ \mathsf{X.Y}{=}X_1,\ \mathsf{X.isa}{=}\mathsf{damage} \right\rangle_{b} \end{array} \right)$$

Figure 1.2: Examples of two elementary semantic molecules (I) for an adjective, $(major)'$, and a noun, $(damage)'$, and a derived semantic molecule (II) obtained by combining them, $(major\ damage)'$.

of others are called *derived semantic molecules* (II).

The head, h, of a semantic molecule is represented as a one-level feature structure (i.e., feature values are atomic). In Figure 1.2 the heads are shown as attribute-value matrices (AVMs). Each molecule has at least two attributes encoding the syntactic category of the associated string, cat, and the semantic head of the string, head. For adjectives, for example, besides these two attributes, there is an attribute, mod, which specifies the semantic index of the modified noun. This information is necessary for combining an adjective and a noun to obtain a noun phrase (e.g., *major damage*). For nouns, we can have other syntactic information (e.g., nr) used for agreement (i.e., number agreement between the subject and the main verb of a sentence). All these sets of attributes are finite and known a priori for each syntactic category. Being a one-level feature structure, no recursive or

embedded structures are allowed, which makes this representation appealing for a learning framework. The recursion is obtained through the recursive grammar rules and the composition constraint, which are described in the next section.

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 et al., 1999)), called OntoSeR (Ontology-based Semantic Representation). It is a logical form built using a set of atomic predicates (APs) based on the concept of attribute-value pair:

$$\begin{aligned}
\langle \text{OntoSeR} \rangle &\stackrel{\text{def}}{=} \langle \text{AP} \rangle \mid \langle \text{OntoSeR} \rangle \wedge \langle \text{OntoSeR} \rangle \\
\langle \text{AP} \rangle &\stackrel{\text{def}}{=} \langle \text{conceptID} \rangle \, . \, \langle \text{attr} \rangle = \langle \text{concept} \rangle \\
\langle \text{concept} \rangle &\stackrel{\text{def}}{=} \langle \text{conceptID} \rangle \mid \langle \text{conceptName} \rangle \\
\langle \text{attr} \rangle &\stackrel{\text{def}}{=} \langle \text{attrID} \rangle \mid \langle \text{attrName} \rangle
\end{aligned}$$

where, $\langle \text{conceptID} \rangle$ is a variable denoting a frame in the ontology, $\langle \text{conceptName} \rangle$ is the name of a frame in the ontology, $\langle \text{attrID} \rangle$ is a variable denoting a slot of a frame, while $\langle \text{attrName} \rangle$ is the name of a slot of a frame in the ontology. The slot is either a property or a relation. As seen in Figure 1.2 (I), our semantic representation is influenced by the ontology-based approach to semantic interpretation. For example, in our framework, the meaning of a noun is the corresponding basic concept in the ontology ($X_3.isa = damage$). The meaning of an adjective is the concept corresponding to a value of a property (slot) of another concept denoted by a noun ($X_1.isa = major, X_2.Y = X_1$), 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 *damage* after the composition that derives *major damage*). The variable

Y will be instantiated during the semantic interpretation on the ontology (e.g., for the noun phrase *major damage*, $Y = degree$).

In this dissertation, we argue that our computational semantics framework has several properties:

1. **Expressive adequacy.** Our representation correctly expresses diverse and complex linguistic phenomena (e.g., coordination, noun-noun compounds, nominalizations, raising/control, long-distance dependencies). The actual semantic representation, OntoSeR, is an ontology query language.

2. **Grammar compatibility.** Semantic representations must be linked to other kinds of grammatical information. The modular design of the semantic molecules allows us to connect them to a constraint-based grammar formalism, where the semantic composition and the semantic interpretation are encoded as grammar constraints. Thus the semantic representation is cleanly linked to syntax, assuring grammar compatibility.

3. **Computational tractability.** Computational tractability requires efficient means to process meanings and to check semantic equivalences, as well as straightforwardly expressing relationships between semantic representations. We use a semantic interpreter which guarantees the soundness and completeness of concept identification.

4. **Reversibility.** The semantic representation, OntoSeR, should contain the information that allows grammar reversibility. The filtering of the information that is not necessary (e.g., for a particular task) is done at the interpretation level. The reversibility is possible giving to the generator only the seman-

tic representation (OntoSeR), without the information from the head of the semantic molecule.

5. **Uniform Representation.** The semantic representation, OntoSeR, must be independent of the knowledge level where the assertion takes place (terminology, factual, discourse). In this way, our semantic interpreter allows the interface with different Knowledge Representation Systems.

6. **Learnability**. Efficient learning requires simple representational devices. Both the head and the body of the semantic molecule are flat representations. Moreover, the modular design and the use of a constraint-based grammar formalism allow the learning of both the grammar rules and the compositional constraints.

The first three criteria are the object of most ongoing research in computational semantics, the tradeoff between expressive adequacy and computational tractability being one of the major issues in semantic-based applications (Copestake et al., 1999). In our work, we are not only interested in computational tractability but also in the *learnability* of the computational semantics framework. Reversibility and uniform representation preserve the entire meaning of utterances at the level of OntoSeR, emphasizing the view of natural language as problem formulation, and not problem solving. Filtering and reasoning can take place at the level of knowledge representation. In other words, at the OntoSeR level we have the entire utterance meaning and nothing more.

1.2.2 Grammar Formalism

This section summarizes the answers to the question "What properties should the grammar have to include semantics and to be learnable?"

Constraint-based grammar frameworks have been widely used to capture both aspects of syntax and semantics. In particular, the Definite Clause Grammar formalism (Pereira and 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, which 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.

However, it is known that classical DCGs are Turing equivalent, and thus undecidable. In this dissertation, we introduce a particular type of DCGs, called *Lexicalized Well-Founded Grammars* which are decidable and learnable. The LWFGs have the following main properties:

- augment the nonterminals with pairs of natural language strings and their *semantic molecules*;
- have 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);
- introduce a *partial ordering among nonterminals*, which allows the ordering of

grammar rules. This partial ordering motivates the naming of our grammar formalism as "well-founded."

The first two properties, facilitated by the DCG-like formalism, allow us to have a grammar that integrates both syntax and semantics. The language generated by the grammar consists of pairs of natural language strings and their semantic molecules. We call these units derived by the grammar *syntagmas*. The compositional constraints are learned together with the grammar rules. The ontology-based constraints are used during parsing to provide access to meaning.

The third property allows the bottom-up learning of LWFGs from a set of ordered *representative examples*. The representative examples are the simplest syntagmas generated by a LWFG. The ordering of examples allows for an efficient bottom-up relational learning of the grammar rules and their compositional semantic constraints. Besides efficiency, a practical advantage of the representative example set is its small size, equal to the cardinality of the grammar rule set.

In Figure 1.3 we show the parse tree for the derived syntagma associated with the string *major damage* (see also Figure 1.2). It can be seen that each nonterminal is augmented with a syntagma (i.e., pair of a string and its semantic molecule).

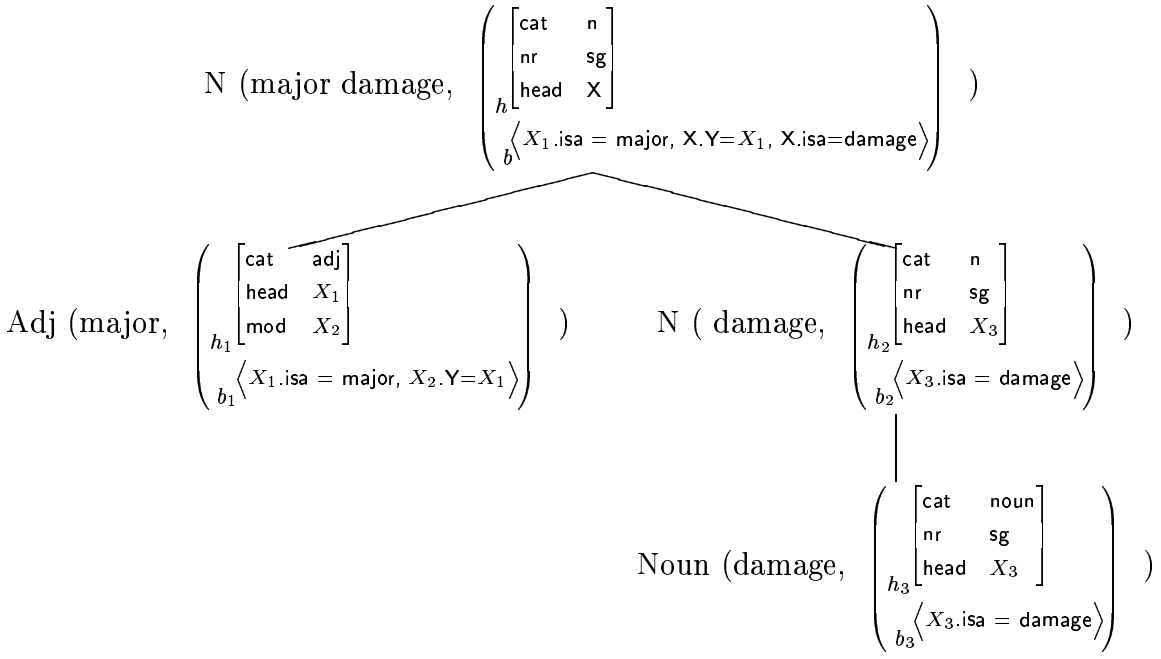
The general form a LWFG rule is given below:[1]

$$A(w, \binom{h}{b}) \rightarrow B_1(w_1, \binom{h_1}{b_1}),\ \ldots, B_n(w_n, \binom{h_n}{b_n}):\ \Phi_{comp}(h, h_1, \ldots h_n),\ \Phi_{onto}(b)$$

where:

- $A, B_1, \ldots, B_n$ are grammar nonterminals, which represent syntactic categories that are given in the head of the semantic molecules ($A = h.cat,\ B_i = h_i.cat$). In Figure 1.3 we have N, Adj, $Noun$ as nonterminals.

[1]For the clarity of the presentation we keep the notation below, and not the DCG notation.



(a)

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

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

(b)

Figure 1.3: Parse tree for syntagma associated with the string *major damage*

- $w, w_1, \ldots, w_n$ are variables for natural language strings. In the sample derivation given in Figure 1.3 they are instantiated with *major damage*, *major*, and *damage*, respectively (n=2).

- $\binom{h}{b}, \binom{h_1}{b_1}, \ldots, \binom{h_n}{b_n}$ are the semantic molecules corresponding to the natural language strings $w, w_1, \ldots, w_n$, respectively.

- : is a delimiter for constraints.

- Φ_{comp} is a semantic composition constraint, which shows how the meaning of the whole is composed from the meaning of its parts. The composition constraint is applied only to the heads of the semantic molecules, the bodies being just concatenated. Φ_{comp} is a system of equations, and is learned together with the grammar rules. An example is given in Figure 1.3 and a full description is given in Chapter 4. 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 in Figure 1.3 are bound). At this point the variable Y is still not instantiated.

- Φ_{onto} is the ontology-based semantic interpretation constraint applied only to the body of the semantic molecule corresponding to the left-hand side nonterminal. Thus, access to meaning is provided at the grammar rule level, which is a source for disambiguation. The ontology-based interpretation is not done during the composition operation, but afterwards. Thus, for example, the head of the noun phrase *major damage* (Figure 1.3) 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. When Φ_{onto} is applied, the variable Y becomes

instantiated with values taken from the ontology (e.g., *degree*).

Thus, semantic composition and semantic interpretation are constraints at the grammar rule level. Moreover, the semantic compositional constraints are learned together with the grammar rules. This is possible due to the determinacy property of these constraints. On the one hand, Φ_{comp} and the syntagmas corresponding to the nonterminals from the right-hand side of the grammar rule, $(w_i, \binom{h_i}{b_i})$, completely determine the syntagma corresponding to the left-hand side nonterminal, $(w, \binom{h}{b})$. On the other hand, $(w, \binom{h}{b})$ and $(w_i, \binom{h_i}{b_i})$ completely determine Φ_{comp}. The latter is relevant when learning Φ_{comp} together with the grammar rule.

Unambiguous language. In our framework, the language generated by the grammar consists of syntagmas (i.e., pairs of strings and their semantic molecules). Thus, when we talk about unambiguity we refer to the syntagmas. Let us consider the classical example of prepositional phrase ambiguity. The utterance *I saw the man with the telescope* has at least two derivations: one where the **PP** *with the telescope* modifies *man*, and one where it modifies *saw*. In our framework, since the language is formed by syntagmas, these ambiguities are eliminated. The utterance *I saw the man with the telescope* has two associated semantic molecules: one where the **PP** is attached to the verb *saw*, and one where it is attached to the noun *man*. Thus, we have two syntagmas which are unambiguous.

From the learning perspective, unambiguity is an assumption. Both the representative examples set and the sublanguage used for generalization must be unambiguous. Since these two sets of examples are in fact syntagmas (pairs of utterances and their semantic molecules), the unambiguity requirement can be fulfilled.

We will describe this issue in detail in Chapter 5.

In Chapter 3 we present the Lexicalized Well-Founded Grammars and we define their denotational and operational semantics, the derivation operation, the parsing/generation mechanism, the representative examples and the representative sublanguage.

1.2.3 Learning Model

This section summarizes the answers to the question "What learning paradigm is needed?"

Inductive Logic Programming (ILP) is a class of relational learning methods which is concerned with inducing first-order Horn clauses from examples and background knowledge. Kietz and Džeroski (1994) describe formally the ILP-problem and relate it to Gold (1967) and PAC-learnability (Valiant, 1984) frameworks. They show that the ILP-problem and Gold's identification in the limit are not strongly connected, but they define PAC-learnability for the ILP setting. We briefly present the ILP learning problem and then discuss how our grammar induction problem is framed in this approach.

The ILP Learning Problem

Given:

- a correct provability relation $\vdash$ for a first-order language L,
- a background knowledge B in language LB: $B \in LB \subseteq L$,
- positive and negative examples $E = E^+ \cup E^-$ in language $LE \subseteq L$, and
- a hypothesis language $LH \subseteq L$.

Find a hypothesis $H \in LH$ such that:

- $H \wedge B$ explains the positive examples E^+, and $H \wedge B$ does not explain the negative examples E^-.

The tuple $\langle \vdash, LB, LE, LH \rangle$ is called the *ILP-learning problem*. The general ILP-learning problem $\langle \models$, *Horn clauses, ground atoms, Horn clauses*$\rangle$ is undecidable. Thus, it is neither consistently identifiable in the limit, nor PAC-learnable. The question is what subclasses of first-order logic are efficiently learnable. Possible choices to restrict the ILP-learning problem are: the provability relation, $\vdash$ (also called the generalization model), 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.

The LWFG Induction Problem in the ILP-setting

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

- The provability relation, $\vdash$, is given by the robust parsing, and we denote it by $\vdash_{rp}$. We use the "parsing as deduction" technique (Pereira and Warren, 1983; Shieber, Schabes, and Pereira, 1995). Using this technique, 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 LWFG induction problem is decidable.

- The language of background knowledge, LB, is the set of LWFG rules (a type of DCG rules) that are already learned, together with the 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 preserves the parsing of representative examples.

That is, our LWFG induction problem can be formulated as $\langle \vdash_{rp}$, *LWFG rules (type of DCG rules) + elementary syntagmas (ground atoms), representative sublanguage (ground atoms), LWFG rules (type of DCG rules)* $\rangle$.

Grammar Induction Model

We have formulated the LWFG induction problem in the ILP-setting. The theoretical learning model is called *Grammar Approximation by Representative Sublanguage (GARS)*, and can be formulated as follows:
Given:

- a representative example set E_R, lexically consistent (allows the construction of the grammar lattice bottom element, $\perp$)

- a finite sublanguage E_σ, conformal and thus unambiguous, which includes the representative example set, $E_\sigma \supseteq E_R$. We call this sublanguage the *Representative Sublanguge*

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

In this dissertation we prove that the search space for LWFG learning is a complete grammar lattice, and we give a learnability theorem for LWFG learning. This learnability result significantly extends the class of problems learnable by ILP methods, with a particular type of DCGs (our Lexicalized Well-Founded Grammars).

1.2.4 Applications of the LWFG Learning Model

This section summarizes the answers to the question "What applications does our language learning model have?"

Our learning model has the advantage of learning a grammar from a small set of semantically annotated examples. However, a quantitative evaluation would require a large semantically annotated treebank (OntoSeR annotation), which would require considerable human effort. Thus, in this dissertation we focus on qualitative evaluations that show the potential applications of our LWFG learning model. For this, we have implemented a general LWFG learning system which fully covers the theoretical GARS model. In addition, we have implemented a particular semantic interpreter for the qualitative evaluation of our learned grammar.

The qualitative evaluations cover the following issues:

- Coverage of diverse and difficult linguistic phenomena such as raising and control constructions, long-distance dependencies, noun compounds, nominalization, coordination, verbal constructions (including tense, aspect, modals, negation), complex noun phrases, and relative clauses.

- Terminological knowledge acquisition from natural language definitions (in the medical domain).

- Handling both precise and vague questions w.r.t. an utterance or set of utterances and obtaining precise answers at the concept level. Since we are concerned only with the evaluation of the grammar learning model, the answers should be explicitly stated in text (i.e., no reasoning).

Since in these applications we do not always deal only with utterances at the grammar level, we need to define two additional levels of representations:

- Text Knowledge Representation (TKR) — the representation of utterance(s) after assertion. TKR is an asserted form of OntoSeR, and it is a representation of the entire text. We still have at this level the reversibility property (i.e., from this representation we can directly obtain the text through our parser/generator).

- Ontology-level Knowledge Representation (OKR) — From TKR we can obtain the OKR representation through filtering and by introducing the concept identity principle, which establishes a bijection between a concept and a referent. We define the OKR as a directed acyclic graph (DAG), where vertices are concepts or instances of concepts, and edges are semantic roles (including properties). In this dissertation we only define and implement a weak concept identity principle.

In order to illustrate the applications that our LWFG learning model has, we give an example of knowledge acquisition/query for constructing a hierarchy of concepts. For this, we used our learned grammar and the implemented semantic interpreter. The definitions and the OKR of these definitions are presented in Figure 1.4. The questions and answers related to this OKR are given in Figure 1.5,

where we give the answer concept together with its defining subDAG. The answer is obtained though a DAG matching algorithm, where the wh-word matches the answer concept.

The acquisition of knowledge can be done directly. We consider both concepts (#hepatitis, #blood) and instances of concepts (#virus25,#virus33) in our OKR representation.

The defined term (definiendum) is always a concept, and it is part of the sub hierarchy (the inverse of the is_a hierarchy). The concepts in the sub hierarchy are presented with double square boxes in Figure 1.4. All the definitional properties of the concepts are directly linked to the concept vertex (facilitated by our interpretation of copula be-predicative). As we can see, linguistic phenomena of theoretical interest to linguists become important in applications. Besides the concepts that are defined, we can also have concepts that are referred (i.e., they are part of the definiens), if they do not have any modification (e.g., #blood in definition of Hepatitis A, and Hepatitis B).

If a referred concept has modifications, it is represented as an instance of a concept in the OKR. As a consequence, different verbalizations of concept properties can be differentiated in OKR, allowing us to obtain direct answers that are specific to each verbalization. For example, the term *virus* appears in the definition of both Hepatitis A and Hepatitis B. In OKR, they are two different instances of a concept, #virus25 and #virus33, since they have different modifications: *persists in the blood serum*, *does not persists in the blood serum*, respectively. These modifications are an essential part of the differentia of concepts #HepatitisA and #HepatitisB, making the distinction between the two. When we ask the question *What is caused*

1. Hepatitis is a disease caused by infectious or toxic agents and characterized by jaundice, fever and liver enlargement.
2. Hepatitis A is an acute but benign viral hepatitis caused by a virus that does not persist in the blood serum.
3. Hepatitis B is an acute viral hepatitis caused by a virus that tends to persist in the blood serum.

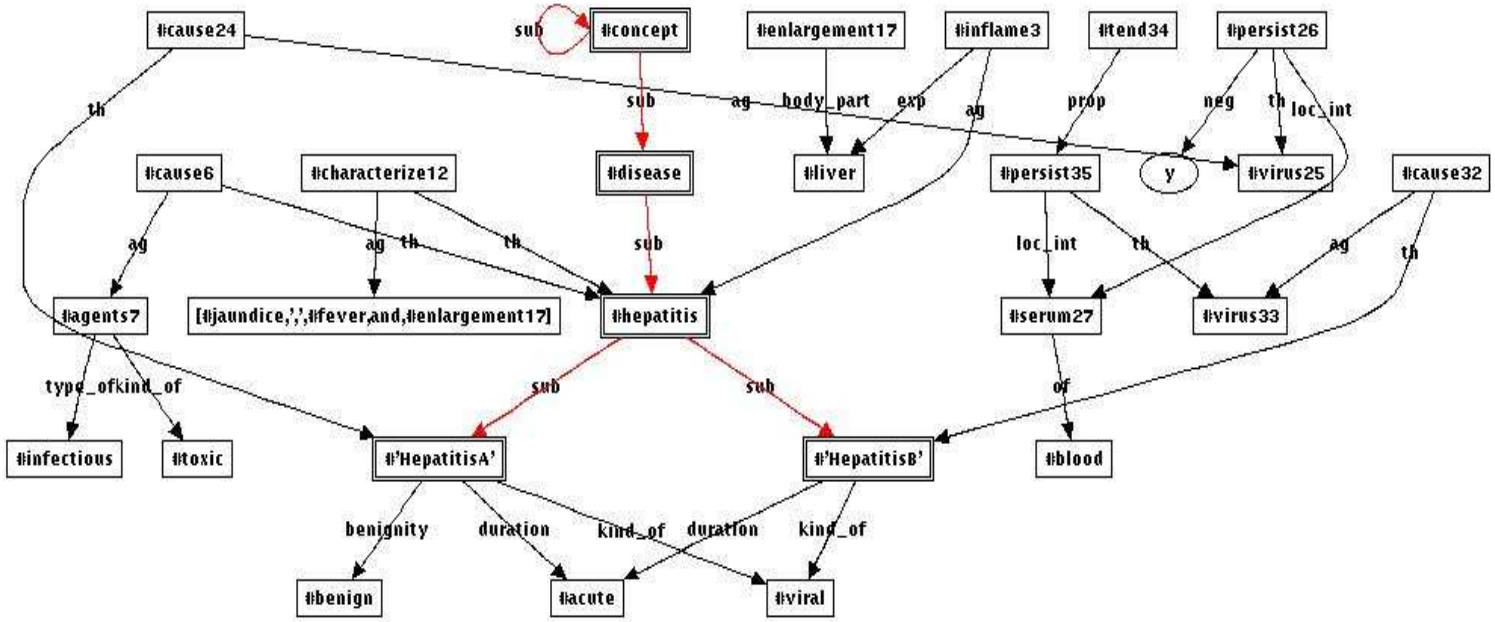
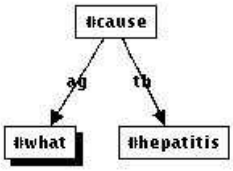


Figure 1.4: OKR for the definitions given in Figure 1.1(b)

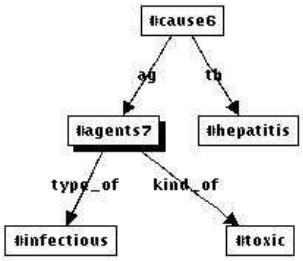
by a virus that persists in the blood serum? (Q3 in Figure 1.5), we obtain only the correct answer #HepatitisB (A3 in Figure 1.5). Our representation facilitates vague questions as well, such as, *What is caused by something that does not persist in the blood serum?* (Q2 in Figure 1.5). This is obtained by considering *something* as a variable concept that matches a vertex in the OKR. A practical advantage is that we can obtain all the concepts that are in a particular relation with other concepts, or that have particular properties.

Another important aspect that contributes to the adequacy of our representation for acquisition and query is the OKR-equivalences that we obtain for different syntactic forms. They are related mainly to verbal constructions. Since we deal with terminology, temporal reasoning is not important, and thus we ignore tense and aspect information (they are filtered at the OKR level). For terminology, however, modals and negation need to be taken into account. For example, negation is essential for differentiating Hepatitis A and Hepatitis B. Thus, we do represent negation (#persist26 has the negation represented). Among OKR-equivalences we have: 1) active and passive constructions; 2) *-ed* and *-ing* verb forms in reduced relative clauses are equivalent to passive/active verbal constructions (e.g., the question can be formulated in present tense, active voice *What causes hepatitis?* (Q1 in Figure 1.5), while the answer is obtained from a definitional statement involving the reduced relative clause *hepatitis is a disease caused by infectious or toxic agents ...* (A1 in Figure 1.5)); 3) constructions involving raising verbs, where we can take advantage that the controller is not the semantic argument of the raising verb (e.g., in the definition of Hepatitis B we have *... caused by a virus that tends to persist in the blood serum*, while the question can be asked without the raising verb *What*

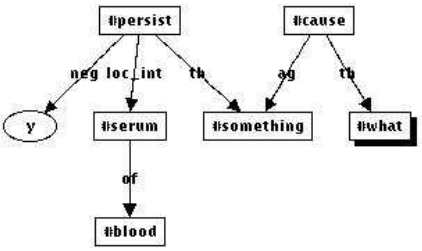
Q1: What causes hepatitis?



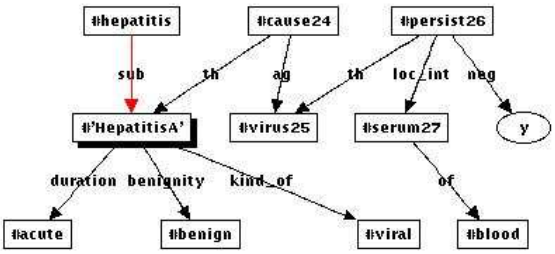
A1: #agents7



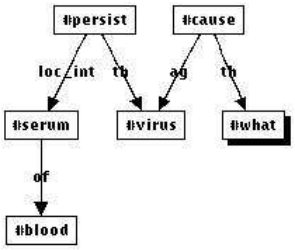
Q2: What is caused by **something** that does not persist in the blood serum?



A2: #'HepatitisA'



Q3: What is caused by a virus that persists in the blood serum?



A3: #'HepatitisB'

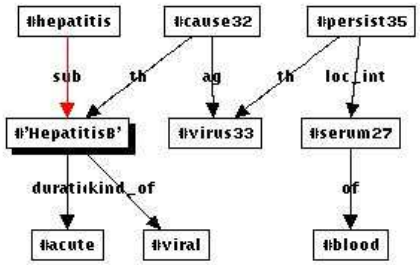


Figure 1.5: OKR for questions and answers

is caused by a virus that persists in the blood serum?).

The ample qualitative evaluations done in this dissertation, some of which we sketched above, are useful in order to develop, in future work, quantitative evaluations of our grammar learning model without a large annotated treebank (at the OntoSeR level), which would require considerable human effort. The application of directly acquiring terminological knowledge from text can be used in the future to enhance OKR with probabilities. This probabilistic enhancement only at the conceptual level bears similarity to Pinker's theory of child language acquisition that there is "a formal and nearly exceptionless grammatical linkage between syntax and semantics, and a more probabilistic cognitive correlation between semantics in parental speech and childlike concepts" (Pinker, 1989, page 364). Our current system requires manual validation of the obtained OKR since the level of ambiguity is still fairly high (see Section 8.4). Adding probabilities at the OKR level will eliminate the need for manual validation, and thus will allow a future quantitative evaluation of our grammar learning model based on question-answering-like experiments on a large set of unannotated utterances. This corpus of utterances used for LWFG learning evaluation should contain precise answers explicitly stated in the text (i.e., without reasoning).

1.3 Contributions

While the research in this dissertation spans several topics, the common thread we follow is the learnability issue. We have grouped our contributions in two categories: theoretical and application-oriented.

Theoretical Contributions

1. **Semantic molecule: a new representation of natural language for learning.** We introduce a new representation that is expressive enough to model complex aspects of natural language and simple enough to be efficiently used in a relational learning framework. The semantic representation, OntoSeR, is directly interpretable on the ontology, it is independent on the knowledge level where the assertion takes place, it ensures learnability, and it is used alone as input to our generator. The semantic composition is encoded as grammar constraints that are learned together with the grammar rules. The semantic interpretation is encoded as grammar constraints as well, providing access to meaning during parsing.

2. **Lexicalized Well-Founded Grammars: a new type of learnable constraint-based grammars.** LWFGs are a learnable type of Definite Clause Grammars. They model syntax and semantics and have constraints at the rule level for semantic composition and semantic interpretation. LWFGs introduce a partial ordering among grammar nonterminals, which allows their bottom-up induction from a set of representative examples and a representative sublanguage. We give the definition of the representative examples set and we provide an efficient algorithm for generating this set, given a grammar and a sublanguage.

3. **Grammar Approximation by Representative Sublanguage: a new computationally efficient model for grammar learning.** In this dissertation we prove that LWFGs are always learnable under the GARS model,

i.e., the learning always converges to the target grammar. We show that the search space is a complete grammar lattice and we provide polynomial algorithms for grammar induction proving their soundness. The learnability theorem extends significantly the class of problems learnable by ILP methods. This class is a class of constraint-based grammars which capture syntax and semantics (LWFG) and which are learnable by relational learning methods.

Application-oriented Contributions

1. **Framework for Grammar Learning and Merging.** In this dissertation, we have implemented a system that represents an experimental platform for all the theoretical algorithms. This system consists of: 1) an Inductive Logic Programming system that learns only from positive examples, uses background knowledge and has a dual mode of operation: learns both from ordered and unordered representative examples; and 2) a robust parser/generator. We have learned an experimental grammar that covers diverse and complex linguistic phenomena, such as raising and control, long-distance dependencies, relative clauses, noun compounds, nominalizations, coordinations, complex verbal constructions. This framework has the practical advantage of implementing sound grammar revision and grammar merging, which allow an incremental coverage of natural language fragments.

2. **Semantic Interpreter For Text-to-Knowledge Acquisition.** We have built a semantic interpreter based on a weak semantic context. Using this interpreter, we have provided qualitative evaluations that show that our semantic representation OntoSeR, and its derived forms TKR and OKR, are

expressive enough to represent all the above-mentioned linguistic phenomena covered by our learned experimental grammar, and are useful for direct terminological knowledge acquisition and querying (concept-level answers to questions). The semantic interpreter implements the weak concept identity principle for terminological knowledge.

The overall scientific finding:

If natural language can be captured by LWFGs, and the set of representative examples and the representative sublanguage used for generalization can be given based on linguistic knowledge, then natural language can be learned.

In this dissertation, we have shown that several fragments of natural language can be represented by LWFGs, but only future research can answer the question: Can LWFGs capture all of natural language?

1.4 Dissertation Guide

In addition to this introductory chapter, the dissertation includes eight chapters and three appendices. **Chapter 2** presents an overview of the related work. The rest of the chapters are organized in two parts.

Part I covers the theoretical aspects of our grammar learning model.

- **Chapter 3** defines the Lexicalized-Well Founded Grammars (LWFGs). We formally define the semantics of LWFGs and give the definition of representative examples, providing an efficient algorithm for generating this set, given a

grammar and a sublanguage. This chapter also gives the definition of robust parsing/generation as deduction.

- **Chapter 4** presents the semantic composition and semantic interpretation as grammar constraints. We describe the properties and principles of our semantic molecule representation. The chapter gives the algorithms for learning the compositional semantic constraints, and presents the semantic interpreter.

- **Chapter 5** describes our theoretical LWFG learning model. We first give the algorithms for grammar learning from ordered and unordered representative examples. We then present how grammar merging can be done in our approach. We give the foundation of the hypothesis search space as Boolean algebra / complete grammar lattice, together with the learnability theorem. Chapter 5 concludes with the presentation of the LWFG induction in the ILP-setting and the description of our Grammar Approximation by Representative Sublanguage (GARS) model.

Part II covers the application-oriented aspects of our grammar learning model.

- **Chapter 6** shows the expressiveness of our semantic representation OntoSeR. We first give the definition of OntoSeR and present the additional levels of representation TKR and OKR. The expressiveness of our representation is shown starting from lexical categories up to complex linguistic phenomena, such as raising and control, wh-questions and relative clauses. We discuss briefly the issue of ambiguity as well. This chapter is one of the qualitative

evaluations of our grammar learning model, as all the presented examples are covered by our learned grammar described in Chapter 7.

- **Chapter 7** presents the general LWFG learning system that implements all the theoretical aspects presented in Part I, in order to learn an experimental grammar whose coverage was discussed in Chapter 6. We present the training data, the overall results and a set of controlled experiments.

- **Chapter 8** shows the usefulness of OntoSeR for direct knowledge acquisition from text, constituting the second qualitative evaluation of our grammar learning model. For this we define rigorously the OKR representation and we introduce the weak concept identity principle. We focus on terminological knowledge acquisition from definitions in the medical domain. We present our implemented semantic interpreter and describe a pilot experiment for the acquisition and querying of a small OKR-annotated treebank.

Chapter 9 gives a summary of the main contributions of this dissertation, discusses the open problems and limitations, and proposes directions for future work.

Chapter 2

Related Work

The general problem of text interpretation involves the determination of the semantic relations among entities and the events they participate in, answering to questions, such as "who" did "what" to "whom", "when", "where", "how" and "why". Work in this area has relied on manually created semantic grammars and other resources for supporting text interpretation (Hirst, 1987; Pustejovsky, 1995; Copestake and Flickinger, 2000). This research has allowed deep language understanding, but has the disadvantage of requiring intensive manual labor, being often restricted to narrow domains. Recently, the compilation of large corpora annotated with semantic roles, such as PropBank (Palmer, Gildea, and Kingsbury, 2005), NomBank (Meyers et al., 2004), and FrameNet (Baker, Fillmore, and Lowe, 1998), has enabled the development of statistical methods for the shallower task of semantic role labeling, especially for labeling of the arguments and adjuncts of verbs (Gildea and Jurafsky, 2002; Chen and Rambow, 2003; Xue and Palmer, 2004; Carreras and Marquez, 2004). The representation introduced in this dissertation, OntoSeR, is an ontology-based semantic representation appropriate for semantic role labeling involving not only verbs and their arguments, but also nouns

and their modifiers, and prepositions and their arguments. The practical use of this representation would have required us to rewrite existing grammars by hand, which is acknowledged to be a daunting task. As a consequence, we define a new type of grammar suitable for language understanding, which is learnable using relational learning methods. Thus, we eliminate the need of rewriting hand-crafted grammars.

In this chapter, we present the work most closely related to the technical aspects of this dissertation, which can be divided into three categories: (1) grammar formalism, (2) natural language representation, and (3) language learning. We focus on discussing the key parts of our framework and how they relate to previous research.

2.1 Grammar Formalism

Constraint-based grammar formalisms[1] have been widely used for natural language understanding and generation. These formalisms include linguistic theories, such as Lexical Functional Grammar (Kaplan and Bresnan, 1982), Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985), and Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994), as well as theory-neutral[2] grammar formalisms which have been developed as computational tools, such as Functional Unification Grammars (FUG) (Kay, 1979), Definite Clause Grammars (DCG) (Pereira and Warren, 1980) and PATR-II (Shieber et al., 1983).

[1] We take the notion of constraint-based from Shieber (1992) where "we characterize the class of formalisms by focusing on their particular use of *information* and *constraints* thereon as an organizing basis. (The formalisms are thus more aptly referred to as information- or constraint-based rather than unification-based)."

[2] By theory-neutral we mean linguistic theory-neutral.

Our Lexicalized Well-Founded Grammars (LWFGs) belong to the second class, being a type of DCGs. An advantage of having a general-purpose formalism is that it allows the linguist to rapidly prototype new linguistic theories and test their behavior computationally. For instance, Shieber (1986) shows how to implement aspects of both LFG and GPSG in the PATR-II formalism. Besides this advantage and unlike DCGs and PATR-II, our LWFGs are learnable, which is an important feature for linguists interested in language acquisition, as well as comprehension and production.

As we mentioned before, LWFGs are defined as a particular type of Definite Clause Grammars (Pereira and Warren, 1980), where the nonterminals are augmented with syntagmas and where we have two types of constraints: one for semantic composition, another for semantic interpretation.

Syntagmas are pairs of strings and their semantic molecules. Seen as form-meaning pairs, syntagmas are similar to the notion of "sign" in HPSG and "construct" in Fluid Construction Grammars[3] (Steels, 2004; Steels, Beule, and Neubauer, 2005). However, the representation is slightly different. Only the head of the semantic molecule is a feature structure. Moreover, our feature structures are flat (i.e., feature values are atomic). The body, which is the string actual semantic representation, OntoSeR, is a flat logic-based representation, where the variables represent concept or attribute identifiers in an ontology. Thus, we have an ontology-based semantics. An important architectural difference between our formalism and both HPSG and FCG, is that we have a context-free "backbone" of rules, and thus

[3]Fluid Construction Grammars are a class of construction grammars. We choose FCG since they have computational implementations, they are reversible, and work has been done in trying to address the issue of learning these types of grammars (Steels, 2004).

we keep the concept of derivation. In both HPSG and FCG there are no rules, but rather a hierarchy/collection of 'constructions', and they do not really have the notion of derivation. In our formalism, the derivation (ground syntagma derivation) allows us to formally define the LWFG semantics.

The semantic composition constraints Φ_{comp} applied to the head of the semantic molecule form a system of equations that is a simplified version of "path equations" (Shieber et al., 1983; van Noord, 1993), as we only deal with flat feature structures. Thus, semantic structures are composed through constraint solving, rather than functional application, which has been used in approaches based on Montague's theory of grammar (Montague, 1973). We have the string concatenation as in PATR II (Shieber et al., 1983) and DCGs, and in addition we have a concatenation of the semantic representations, followed by a variable substitution resulting from the application of the semantic composition constraint Φ_{comp}.

The main novel feature of our formalism is the partial ordering among non-terminals, i.e., the "well-foundedness," which together with our flat representation, makes our formalism decidable and learnable. The learnability property sets our formalism apart from most of the constraint-based formalisms mentioned above. Research on Fluid Construction Grammars has considered the issue of grammar learning, but the setting is completely different: a constructivist approach to language learning through "situated language games played by embodied artificial agents... semantic and syntactic categories as well as the way they are used in constructions is built up in a gradual development process, starting from quite specific verb-island constructions" (Steels, 2004). In our framework the categories are a-priori given, and we learn the grammar rules and the semantic composition

constraints from representative examples using Inductive Logic Programming.

Lexicalized Well-Founded Grammars are reversible and we define parsing and generation as "deduction" (Pereira and Warren, 1983; Shieber, Schabes, and Pereira, 1995). We show that our parser/generator is "efficiently r-reversible" (Neumann and van Noord, 1994).

As we mentioned before, our formalism is a computational formalism and not a linguistic theory. However, concepts and principles from linguistic theories, such as HPSG (Pollard and Sag, 1994), can be incorporated. For example, in this dissertation, we adapted principles of HPSG to formulate the properties and principles of our semantic molecule representation and compositional constraints Φ_{comp}. It is our hope that in the future linguists will use this formalism to prototype their theories. In Part II of this dissertation we have shown how some of the linguistic phenomena that are usually considered as benchmarks for linguistic theories can be modeled in our framework (e.g., long-distance dependencies, raising and control).

2.2 Natural Language Representation

Our semantic molecule is a flat representation. The head is a one-level feature structure (i.e., feature values are atomic). This is different from the feature structures used in other constraint-based formalisms, such as HPSG and LFG (Kaplan and Bresnan, 1982; Bresnan, 2001). The flat feature structure is used by the compositional constraints, Φ_{comp}, which are learned together with the grammar rules. The body of our semantic molecule, OntoSeR, is also a flat representation. In this regard, it is similar to other flat semantic representations, such as Hobbs's logical form (Hobbs, 1985), Minimal Recursion Semantics (Copestake et al., 1999; Copes-

take et al., 2001), and Logical Form (Moldovan and Rus, 2001; Rus, 2002). Unlike these representations, OntoSeR is an ontology-query language where the variables are either concept (frame) or attribute (slot) identifiers in an ontology. It allows a direct transformation into an ontological representation similar to conceptual graphs (Sowa, 1999). OntoSeR alone, without the information from the head of the semantic molecules, is used as input to the generator, in order to obtain natural language strings. The main feature of our semantic molecule representation (both the head and the body OntoSeR) is that it is suitable for learning.

2.3 Grammar Learning

There are two separate lines of research in language learning that our work relates to. First, there is the work in formal grammar induction which aims at proving the learnability of classes of languages (most notably regular and context-free languages) (Angluin, 1982; Sakakibara, 1992; Brazma, 1993; Dupont, Miclet, and Vidal, 1994; Sakakibara, 1997; Seki and Kobayashi, 2004). In this dissertation, we prove the learnability of LWFGs, which extend the class of Context-Free Grammars, opening the way for studying the learnability of more complex classes of grammars. 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 (Brazma, 1993; Freivalds, Kinber, and Wiehagen, 1993; Lange, Nessel, and Wiehage, 1998). In this dissertation, we formally define the representative examples of a Lexicalized Well-Founded Grammar and propose a model of learning LWFGs from representative examples and a representative sublanguage, which we call the Grammar Approximation by Representative Sublanguage (GARS).

Second, there is the work in learning grammatical aspects of natural language. Most research has been focused on syntax (Collins, 1999; Hwa, 1999; Osborne, 1999; Clark, 2001; Klein and Manning, 2001), but recently, there is a growing interest in learning semantics beyond lexical semantics, such as semantic role labeling (Gildea and Jurafsky, 2002; Chen and Rambow, 2003; Xue and Palmer, 2004; Carreras and Marquez, 2004), or learning to map from text to logical forms for natural language interfaces to databases (Zelle and Mooney, 1993; Tang and Mooney, 2000; Zettlemoyer and Collins, 2005). In our work, we are interested in learning grammars that interleave syntax and semantics, and use an ontology to provide access to meaning during parsing. As we mentioned before, we learn from a set of representative examples, which is a different type of training data. First, the representative examples are not only entire sentences, but they contain smaller units corresponding to words, phrases, clauses or sentences. Second, the type of annotation (i.e., our semantic molecule) is different from the ones used in the previously mentioned learning methods. Unlike Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1994) and PropBank (Palmer, Gildea, and Kingsbury, 2005), we do not have the entire syntactic derivation present in the annotation. Unlike the logical form annotation used for NLIDB (Tang and Mooney, 2000; Zettlemoyer and Collins, 2005), we have both the semantic representation (the body of the semantic molecules) and information about categories (the head of the semantic molecules).

In this dissertation, we use a relational learning method for our grammar induction, based on Inductive Logic Programming (ILP). Kietz and Džeroski (1994) describe formally the ILP-problem and relate it to Gold (1967) and PAC-learnability (Valiant, 1984) frameworks. 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. In this dissertation, we have defined the LWFG learning as a decidable Inductive Logic Programming problem. The decidability of ILP is a consequence of LWFG decidability, which is guaranteed by the fact that we chose as provability relation of ILP, the robust parsing provability. We use a particular type of ILP method, called Inverse Entailment (Muggleton, 1995). Unlike Muggleton, our GARS model does not learn from random examples, but from representative examples (positive only), using as performance criteria the reduced semantics to a representative sublanguage. Unlike other ILP methods, including Muggleton's Inverse Entailment, we prove that for LWFG learning the search space is a complete grammar lattice/Boolean algebra, which guarantees the uniqueness of the solution, and allows us to provide polynomial learning algorithms. The learnability theorem that we give in this dissertation extends significantly the class of problems learnable by ILP methods.

Part I

Theoretical Part

Chapter 3

Lexicalized Well-Founded Grammars

In this chapter we introduce our new type of constraint-based grammars, *Lexicalized-Well Founded Grammars (LWFGs)*. The research presented in this chapter is based on our papers (Muresan, 2004; Muresan, Muresan, and Klavans, 2004; Muresan, Muresan, and Klavans, 2005). LWFGs are a type of Definite Clause Grammars (Pereira and Warren, 1980), where

- the Context-Free Grammar backbone is extended by introducing a partial ordering relation among nonterminals (Well-Founded Grammars);
- the nonterminals are augmented with syntagmas (i.e., pairs of strings and their semantic molecules);
- grammar rules have two types of constraints — one for semantic composition and one for semantic interpretation.

In Section 3.1 we define the Well-Founded Grammars, and give an algorithm to verify if a CFG is a WFG. In Section 3.2 we show how Well-Founded Grammars

are augmented with semantics, and we give the definition of the Lexicalized Well-Founded Grammars. Section 3.3 presents the derivation in LWFGs and Section 3.4 defines the operational and denotational semantics of LWFGs. In Section 3.5 we give the definition of robust parsing/generation as deduction. We conclude this chapter by defining the representative examples and representative sublanguage.

3.1 Well-Founded Grammars

Well-Founded Grammars extend the 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 among the nonterminals (we use the notation* $\succeq$*).*

(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] $\overset{*}{\Rightarrow}$*, is defined as:*

$\frac{A \to w}{A \overset{*}{\Rightarrow} w}$ *(if* A *is a grammar preterminal, i.e.,* $w \in \Sigma$*), and*

$\frac{B_i \overset{*}{\Rightarrow} w_i,\ i=1,\ldots,n, \quad A \to B_1,\ldots,B_n}{A \overset{*}{\Rightarrow} w}$*, where* $w = w_1 \cdots w_n$.

The *language* of a grammar G is $L(G) = \{w | w \in \Sigma^+, S \overset{*}{\Rightarrow} w\}$. The set of all strings generated by a grammar G is $L_s(G) = \{w | w \in \Sigma^+, \exists A \in N_G, A \overset{*}{\Rightarrow} w\}$. 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 $L_s(A) = \{w | w \in \Sigma^+, A \in N_G, A \overset{*}{\Rightarrow} w\}$, and the set of strings generated by *a rule* $A \to \beta$ of the grammar G is $L_s(A \to \beta) = \{w | w \in \Sigma^+, (A \to \beta) \overset{*}{\Rightarrow} w\}$.[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 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 complexity of the algorithm is $O(|P_G|^2 * |\beta|)$.[5]

A nonterminal is assigned to a level when it appears on the left-hand side of an ordered non-recursive rule (Figure 3.1). 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

[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 $(A \to \beta) \overset{*}{\Rightarrow} w$ to denote the derivation $A \overset{*}{\Rightarrow} w$ obtained using the rule $A \to \beta$ in the last derivation step.

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

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 $\emptyset$

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 3.1 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 3.1(b)). We use Rc to denote relative clauses.

Based on this partial ordering relation and applying a topological sorting algorithm we obtain a total ordering among the grammar nonterminals. We use the same notation $\succeq$ for the total ordering relation. In Section 3.6 we show that

(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 set	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 3.1: (a) Grammar, G; (b) Iterative steps for the Well_Founded_Grammar(G) algorithm

this order among nonterminals in Well-Founded Grammars provides a total ordering among the strings derived by these grammars.

3.2 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 associate structures to nonterminals and to add constraints at the grammar rule level.

In 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 Definite Clause Grammars, *Lexicalized Well-Founded Grammars*.

3.2.1 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, $w' = \binom{h}{b}$, 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 3.2 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).[6] We will describe the composition operation, $\circ$, which combines several semantic molecules to form a derived semantic molecule in the next section.

The head, h, of a semantic molecule is represented as a one-level feature structure (i.e., feature values are atomic). In Figure 3.2 the heads are shown as attribute-value matrices (AVMs). Let $\mathcal{A}_h$ be the set of attributes of a molecule

[6]For readability reasons, we sometimes use in figures, or in the running text, the notation $h \bowtie b$ to exemplify the semantic molecules, instead of $\binom{h}{b}$ notation. For example, the molecules in Figure 3.2 are the same as the ones presented in Introduction in Figure 1.2 (page 7).

I. Elementary Semantic Molecules

I – 1 $(major/adj)' = h_1 \bowtie b_1$

$$= \begin{bmatrix} cat & adj \\ head & X_1 \\ mod & X_2 \end{bmatrix} \bowtie \langle X_1.isa = major, X_2.Y = X_1 \rangle$$

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

I – 2 $(damage/noun)' = h_2 \bowtie b_2$

$$= \begin{bmatrix} cat & n \\ nr & sg \\ head & X_3 \end{bmatrix} \bowtie \langle X_3.isa = damage \rangle$$

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

II. Derived Semantic Molecule

$(major \ \ damage)' = h \bowtie b = (major)' \circ (damage)'$

$$= \begin{bmatrix} cat & n \\ nr & sg \\ head & X \end{bmatrix} \bowtie \langle X_1.isa = major, X.Y = X_1, X.isa = damage \rangle$$

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

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 3.2: Augmenting WFG with semantics. Examples of two elementary semantic molecules for an adjective (I-1: $(major)'$) and a noun (I-2: $(damage)'$), and a derived semantic molecule obtained by combining them (II: $(major\ damage)'$). In (III) is given the constraint grammar rule used to derive the string $w = major\ damage$ together with its semantic representation $w' = (major\ damage)'$. The compositional semantic constraint, Φ_{comp}, together with the variable and contextual constant substitutions, ν, and μ, are also shown.

head, h. Each molecule has at least two attributes, one encoding the syntactic category of the associated string, cat, and the other encoding the semantic 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., *major damage*). 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 *major* (adjective) and *damage* (noun), respectively. The set of logical variables of the head, h, is denoted by $var(h)$. Being a one-level feature structure, no recursive or embedded structures are allowed, which makes this representation appealing for a learning framework. Recursion in the grammar is obtained through the recursive grammar rules and the composition constraint, which are described in the next section.

The body, b, of a semantic molecule is a flat representation, called OntoSeR (Ontology-based Semantic Representation). No embedding of predicates is allowed, as in Minimal Recursion Semantics (Copestake et al., 1999). It is a logical form built using a set of atomic predicates (APs) based on the concept of attribute-value pair:

$$
\begin{aligned}
(3.1) \qquad \langle \text{OntoSeR} \rangle &\stackrel{\text{def}}{=} \langle \text{AP} \rangle \mid \langle \text{OntoSeR} \rangle \, \langle \text{lop} \rangle \, \langle \text{OntoSeR} \rangle \\
\langle \text{AP} \rangle &\stackrel{\text{def}}{=} \langle \text{conceptID} \rangle \; . \; \langle \text{attr} \rangle \, \texttt{=} \, \langle \text{concept} \rangle \\
\langle \text{concept} \rangle &\stackrel{\text{def}}{=} \langle \text{conceptID} \rangle \mid \langle \text{conceptName} \rangle \\
\langle \text{attr} \rangle &\stackrel{\text{def}}{=} \langle \text{attrID} \rangle \mid \langle \text{attrName} \rangle
\end{aligned}
$$

where, lop is the logical operator, which we consider in this dissertation to be the logical conjunction ($\wedge$); $\langle$conceptID$\rangle$ is a variable denoting a frame in the ontology; $\langle$conceptName$\rangle$ is the name of a frame in the ontology; $\langle$attrID$\rangle$ is a variable denoting a slot of a frame; and $\langle$attrName$\rangle$ is the name of a slot of a frame in the ontology. The slot is either a property or a relation. The full definition of OntoSeR is given in Chapter 6. As seen in Figure 3.2 (I), our semantic representation is influenced by the ontology-based approach to semantic interpretation. For example, in our framework, the meaning of a noun is the corresponding basic concept in the ontology ($X_3.isa = damage$). The meaning of an adjective is the concept corresponding to a value of a property (slot) of another concept denoted by a noun ($X_1.isa = major, X_2.Y = X_1$). 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 *damage* after the composition that derives *major damage*), while the variable Y will be instantiated during the semantic interpretation on the ontology (e.g., for the noun phrase *major damage*, $Y = degree$).

The semantic composition and the semantic interpretation are introduced in the next section and discussed at length in Chapter 4.

3.2.2 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 and 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. It refers to words, phrases, clauses and sentences.*

The nonterminals of a constraint grammar rule are augmented with generalized syntagmas, the grammar rules having the following form:[7]

[7]For the clarity of the presentation we use the notation below and not the DCG notation.

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

where:

- $A, B_1, \ldots, B_n$ are grammar nonterminals, which represent syntactic categories ($A = h.cat,\ B_i = h_i.cat$).
- $w, w_1, \ldots, w_n$ are variables for natural language strings.
- $w' = \binom{h}{b}, w'_1 = \binom{h_1}{b_1}, \ldots, w'_n = \binom{h_n}{b_n}$ are semantic molecules corresponding to the natural language strings $w, w_1, ...,$ and w_n, respectively.
- : is the delimiter for constraints.
- $\circ$ is the composition of semantic molecules: $b = [b_1, \ldots, b_n]\nu, \Phi_{comp}(h, h_1, ..., h_n)$, where

  $b = [b_1, \ldots, b_n]$ is the concatenation of semantic molecule bodies

  $\Phi_{comp}(h, h_1, \ldots, h_n) = [(h \cup \bigcup_{1 \leq i \leq n} h_i)\mu]\nu$ (see Chapter 4).
- ν, μ are variable and contextual constant substitutions (see Chapter 4).
- $\Phi_{onto}(b)$ - ontology-based semantic interpretation constraint applied only to the body of the semantic molecule corresponding to the left-hand side nonterminal (see Chapter 4).

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.1 and Appendix A.4.2).

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 "$\circ$", 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 3.2.

3.2.3 Lexicalized Well-Founded Grammars: Definition

Lexicalized Well-Founded Grammars augment the Well-Founded Grammars with semantics. These grammars generate languages that consist 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 Section 3.2.2.*
- *The rules (the representation and the constraints) ensure grammar reversibility (see Section 3.5).*

(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 $w' = \binom{h}{b}$ is the semantic molecule of w.

3.3 Derivation in LWFGs

Definition 6. *Given a Lexicalized Well-Founded Grammar G, the ground syntagma derivation, '$\stackrel{*}{\Rightarrow}$',*[10] *is defined as:*

(i) $\frac{A \to \sigma}{A \stackrel{*}{\Rightarrow} \sigma}$ *(if $\sigma = (w, w'), w \in \Sigma, w' \in \Sigma'$, i.e. $A \in N_G$ is a preterminal), and*

(ii) $\frac{B_i \stackrel{*}{\Rightarrow} \sigma_i,\ i=1,\ldots,n, \quad A(\sigma) \to B_1(\sigma_1),\ldots,B_n(\sigma_n):\ \Phi(\bar{\sigma}),\ \bar{\sigma}=(\sigma,\sigma_1,\ldots,\sigma_n)}{A \stackrel{*}{\Rightarrow} \sigma}$

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 *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 $L_\sigma(G) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, s.t. \exists A \in$

[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.

$N_G, A \stackrel{*}{\Rightarrow} \sigma\}$. For a LWFG 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* $A \rightarrow \beta : \Phi$ of the grammar G is $L_\sigma(A \rightarrow \beta : \Phi) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, (A \rightarrow \beta : \Phi) \stackrel{*}{\Rightarrow} \sigma\}$.[12]

3.4 Semantics of LWFGs

Operational Semantics. 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, $L_\sigma(G)$. In Section 3.5, following the insight of "parsing as deduction" (Shieber, Schabes, and Pereira, 1995), we show that a deductive system for parsing LWFGs can serve as a method for defining their operational semantics. That is $G \vdash A(\sigma)$ iff $\sigma \in L_\sigma(G)$.

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

Definition 7. *Let $I \subseteq L_\sigma(G)$ be a subset of syntagmas generated by the LWFG G. We define the immediate syntagma derivation operator $T_G : 2^{L_\sigma(G)} \rightarrow 2^{L_\sigma(G)}$, s.t.: $T_G(I) = \{\sigma \in L_\sigma(G) |$ if $(A(\sigma) \rightarrow B_1(\sigma_1), \ldots, B_n(\sigma_n) : \Phi(\bar{\sigma})) \in P_G \ \wedge \ B_i \stackrel{*}{\Rightarrow} \sigma_i \ \wedge$*

[11] In the remainder of this dissertation 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 $(A \rightarrow \beta : \Phi) \stackrel{*}{\Rightarrow} \sigma$ to denote the derivation $A \stackrel{*}{\Rightarrow} \sigma$ obtained using the rule $A \rightarrow \beta : \Phi$ in the last derivation step. We want to remind our reader that we define the ground derivation bottom-up.

$\sigma_i \in I$ then $A \stackrel{*}{\Rightarrow} \sigma\}$. If we denote $T_G \uparrow 0 = \emptyset$ and $T_G \uparrow (i+1) = T_G(T_G \uparrow i), i \geq 0$, 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 and Kowalski, 1976; Denecker, Bruynooghe, and 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 and Kowalski, 1976). We have $lfp(T_G) = T_G \uparrow \omega$, where ω is the minimum limit ordinal. Thus, the denotational semantics of a grammar G can be seen as the least fixpoint of the immediate 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 5.1.1).

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

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, $mgdl(A \rightarrow \beta : \Phi)$. These concepts are crucial in defining the representative example set of a LWFG, G.

Definition 8. *Let σ be a syntagma.*

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

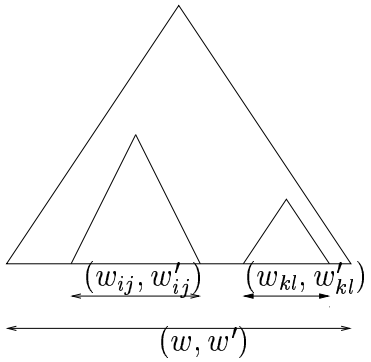


Figure 3.3: Robust parsing: returns all the chunks $(w_{ij}, w'_{ij}), (w_{kl}, w'_{kl})$ of a given syntagma $(w, w\prime)$.

3.5 Parser/Generator for LWFGs

Lexicalized Well-Founded Grammars are reversible grammars. In this section we discuss the parser/generator that is an *effectively r-reversible* program (Neumann and van Noord, 1994), i.e., it is capable of both parsing and generation and it is guaranteed to terminate. It is a bottom-up active chart parser (Kay, 1973).

The semantics of our parser/generator is given by the 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, $w' = \binom{h}{b}$ is its semantic molecule, and $b = b_1 \cdots b_n$ is the string semantic representation.

Definition 9 (Parsing). *We define the set of syntagmas returned by the robust parser by:* $L_\sigma(w) = \{\sigma | \sigma = (w_{ij}, w'_{ij}),\ with\ w_{ij} = w_i w_{i+1} \cdots w_{j-1},\ w_i, w_{j-1} \in \Sigma,\ 1 \leq i \leq j \leq n+1, s.t.\ \exists A \in N_G,\ A \stackrel{*G}{\Rightarrow} \sigma\}$

Definition 10 (Generation). *We define the set of syntagmas returned by the generator by:* $L_\sigma(b) = \{\sigma | \sigma = (w_{ij},\ w'_{ij}),\ with\ w'_{ij} = \binom{h_{ij}}{b_{ij}},\ b_{ij} = (b_i b_{i+1} \cdots b_{j-1})\nu_{ij}, w'_i = \binom{h_i}{b_i} \in \Sigma'\ 1 \leq i \leq j \leq n+1,\ s.t.\ \exists A \in N_G,\ A \stackrel{*G}{\Rightarrow} \sigma\}$

Definition 11 (Parsing during learning). *When both the string (w) and its semantic representation (b) are given, we define the set of syntagmas parsed by*

the robust parser by $L_\sigma(w,b) = \{\sigma | \sigma = (w_{ij}, w'_{ij}), with\ w_{ij} = w_i \cdots w_{j-1}, w'_{ij} = \binom{h_{ij}}{b_{ij}}, b_{ij} = (b_i b_{i+1} \cdots b_{j-1})\nu_{ij},\ w_i \in \Sigma,\ w'_i = \binom{h_i}{b_i} \in \Sigma',\ 1 \leq i \leq j \leq n+1,\ s.t.\ \exists A \in N_G,\ A \stackrel{*G}{\Rightarrow} \sigma\}$

It can be noticed that the parser/generator returns all the subsyntagmas, and thus we said that it is robust (produces all the chunks, not only the full parse, see Figure 3.3).

In general, for a given syntagma $\sigma = (w, \binom{h}{b})$ 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 A.3).

We assume that $T_G(\emptyset)$ is given, that is the rules for preterminals are given: $\sigma \in T_G(\emptyset)$ iff $\sigma = (w, w') \in \Sigma \times \Sigma' \wedge B \in N_G \wedge B \rightarrow \sigma \in P_G$.

3.5.1 Parsing/Generation as Deduction

Parsing can be viewed as a deductive process that tries to prove claims about the grammaticality of a string from assumptions about the grammatical status of the string's elements and the linear order between them (Pereira and Warren, 1983). Several benefits have been claimed about the association between parsing and deduction: 1) existing logics can be used as a basis for new grammar formalisms with desirable representational and computational properties; 2) the modular separation of parsing into a logic of grammaticality claims and a proof search procedure allows the investigation of a wide range of parsing algorithms for existing grammar formalisms by choosing specific classes of grammaticality claims and specific search procedures (Shieber, Schabes, and Pereira, 1995).

In this dissertation, "parsing as deduction" is useful for providing a formal

definition of ground syntagma derivation, for proving the decidability of LWFGs, and thus for proving their polynomial learnability (issue discussed at length in Chapter 5). We begin by giving the basic notions of parsing as deduction as given in (Shieber, Schabes, and Pereira, 1995).

3.5.1.1 Basic Notions

Shieber, Schabes and Pereira (1995) define parsing as a deductive process in which rules of inference are used to derive statements about the grammatical status of strings from other statements represented by formulas in a suitable formal language. A grammatical deductive system consists of a set of inference rules and a set of axioms and goals given by an appropriate set of formula schemata, $\langle deductive\ system \rangle \stackrel{\text{def}}{=} \langle \langle formula\ schemata \rangle, \langle axioms \rangle, \langle inference\ rule \rangle, \langle goals \rangle \rangle$, where $\langle axioms \rangle \cup \langle goals \rangle \subseteq \langle formula\ schemata \rangle$.

The general form of a rule of inference is

$$\frac{A_1 \quad \cdots \quad A_k}{B} \qquad \langle \text{side conditions on } A_1, \ldots, A_k, B \rangle$$

The antecedents $A_1, \ldots, A_k$ and the consequent B are formula schemata. That means they contain metavariables that will be instantiated with appropriate terms when the inference rule is used. A derivation of a formula B from assumptions $A_1, \ldots, A_m$ is a sequence of formulas $S_1, \ldots, S_n$ such that $B = S_n$ and each S_i is either an axiom (i.e., one of the A_j), or there is a rule of inference R and formulas $S_{i_1}, \ldots, S_{i_k}$ with $i_1, \ldots, i_k < i$ such that for appropriate substitutions of terms for metavariables in R, the formulas $S_{i_1}, \ldots, S_{i_k}$ match the antecedents of the rule, S_i matches the consequent, and the side conditions are satisfied. The notation $A_1, \ldots, A_m \vdash B$ is used, and it means that B is a consequence of $A_1, \ldots, A_m$ if such

a derivation exists. If B is a consequence of the empty set of assumptions, it is said to be derivable, $\vdash B$.

In the parsing-as-deduction framework, the side conditions refer to rules of a particular grammar, and the formulas refer to string positions in the fixed string to be parsed $w = w_1 \cdots w_n$. With respect to the given string, the goal formula states that the string is grammatical according to the given grammar. Then parsing a string corresponds to finding a derivation reaching a goal formula.

We will use standard notation for metavariables ranging over the objects under discussion:[13] $A, B, C, \ldots$ for arbitrary formula or symbols such as grammar nonterminals; $i, j, k, \ldots$ for indices into various strings, especially the string w; $\alpha, \beta, \gamma, \ldots$ for strings of terminal and nonterminal symbols. Substrings will be notated $w_{ij} = w_i, \ldots, w_{j-1}$ for the i-th through $j-1$-th elements of w, inclusive.

3.5.1.2 Robust Bottom-up Active Chart Parser/Generator

In this section the robust bottom-up active chart parsing algorithm (Kay, 1973) for Lexicalized Well-Founded Grammars is presented as a deductive system. The key new feature is that we obtain syntagmas as result of parsing/generation, i.e., we obtain semantic molecules (type of feature structures) associated with the strings.

We assume we have a LWFG $G = \langle \Sigma, \Sigma', N_G, R_G, P_G, S \rangle$, and a syntagma $\sigma = (w, \binom{h}{b})$. For parsing we are given the string w, while for generation we are given the semantic representation b.[14]

Let's consider $\sigma = \sigma_1 \cdots \sigma_n$, where $\sigma_i = (w_i, \binom{h_i}{b_i})$, $1 \leq i \leq n$, $w_i \in \Sigma$ are

[13] Specific notations required for our grammar formalism and parsing algorithm will be introduced in the next subsection.

[14] We mention that our semantic representation OntoSeR alone (without the semantic molecule head) allows string generation.

lexical items, $\binom{h_i}{b_i} \in \Sigma'$ are their corresponding elementary semantic molecules, $w = w_1 \cdots w_n$ (string concatenation), $b = (b_1 \cdots b_n)\nu$ (semantic representation concatenation followed by variable substitution ν).

To define the deductive systems, we use the following specific notations in addition to the standard notations presented above: $\sigma_{ij}^R = (w_{ij}^R, \binom{h_{ij}^R}{b_{ij}^R})$ are syntagmas corresponding to the partially parsed right-hand side of a rule; $\sigma_{ij}^L = (w_{ij}^L, \binom{h_{ij}^L}{b_{ij}^L})$ are the ground derived syntagmas (i.e., appears in the left-hand side of grammar rules).

The *items* of the logic (i.e., the parsing formula schemata) are of the form $[i, j, \sigma_{ij}, A \to \alpha \bullet \beta \Phi_A]$, where $A \to \alpha\beta \colon \Phi_A$ is a grammar rule, constraint Φ_A can be true, $\bullet$ shows how much of the right-hand side of the rule has been recognized so far, i points to the parent node where the rule was invoked, and j points to the position in the input that the recognition has reached. We can notice here that syntagmas are associated with each item. For elements of the lexicon, the item $[i, i+1, \sigma_{ii+1}^L, B_i \to \bullet]$ makes a true claim, so that such items are taken as axiomatic. The goal items are of the form $[i, j, \sigma_{ij}^L, A \to \alpha \Phi_A \bullet]$, where σ_{ij}^L is ground derived from the rule $A \to \alpha \colon \Phi_A$, i.e., it belongs to the left-hand side nonterminal. Since we considered the syntagmas derived from all grammar nonterminals, not only the start symbol, we have a robust parsing mechanism.

In Kay's terminology, the items are edges. The axioms and goals are inactive edges having $\bullet$ at the end. The rest are active edges.

We have three inference rules used to obtain the goal items from the axioms:

- **Prediction Rule:** This is the bottom-up prediction rule from Kay's bottom-up active chart parsing algorithm, where an empty active arc is added. From an item $[i, j, \sigma_{ij}^L, B \to \beta \Phi_B \bullet]$ (inactive edge) and a grammar rule $A \to B\gamma \colon \Phi_A$

as side condition, the item $[i, i, \sigma_{ii}^R, A \rightarrow \bullet B\gamma\Phi_A]$ (active edge) can be predicted. That is, given that a syntagma σ_{ij}^L has been already ground-derived from a nonterminal B, and given there is a grammar rule $A \rightarrow B\gamma$: Φ_A, we predict the item $[i, i, \sigma_{ii}^R, A \rightarrow \bullet B\gamma\Phi_A]$ having the empty syntagma $\sigma_{ii}^R = (w_{ii}^R, \binom{h_{ii}^R}{b_{ii}^R})$ associated with it, where $w_{ii}^R = \epsilon$ is the empty string, $b_{ii}^R = true$ is the empty semantic representation, and $h_{ii}^R = \emptyset$ is the empty head.

- **Completion Rule:** This rule corresponds to shifting the $\bullet$ across a nonterminal in the right-hand side of the rule. It combines an active and inactive edge to obtain an active edge. Thus, it has two antecedents $[i, j, \sigma_{ij}^R, A \rightarrow \alpha\bullet B\gamma\Phi_A]$ and $[j, k, \sigma_{jk}^L, B \rightarrow \beta\Phi_B\bullet]$, and one consequent $[i, k, \sigma_{ik}^R, A \rightarrow \alpha B\bullet\gamma\Phi_A]$. Shifting the $\bullet$ during the processing of the right-hand side of a rule means string concatenation, conjunction of semantic molecule bodies (i.e., semantic representation) and union of semantic molecule heads. That is, $\sigma_{ik}^R = \sigma_{ij}^R \circ \sigma_{jk}^L$, where $w_{ik}^R = w_{ij}^R w_{jk}^L$ (string concatenation), $b_{ik}^R = b_{ij}^R b_{jk}^L$ (body logical conjunction), $h_{ik}^R = h_{ij}^R \cup h_{jk}^L$ (head union). Figure 3.4 shows the inactive and active arcs corresponding to the completion rule.

- **Constraint Rule:** This rule is the only rule that obtains an inactive edge, from an active edge by executing the grammar constraint Φ_A (the $\bullet$ is shift across the constraint). The execution of the constraint is a SLD resolution $\vdash \Phi_A$ (side condition).[15] By applying the constraint rule as the last inference rule we obtain the ground-derived syntagmas σ_{ij}^L. Thus, the goal items are obtained only after the constraint rule is applied. During this inference rule we have that $\sigma_{ij}^L = \phi(\sigma_{ij}^R)$, where ϕ is defined by: $w_{ij}^L = w_{ij}^R$, $b_{ij}^L = b_{ij}^R \nu_{ij}$, and

[15] SLD resolution is the resolution used in logic programming (van Emden and Kowalski, 1976).

Item form	$[i, j, \sigma_{ij}, A \rightarrow \alpha \bullet \beta \Phi_A]$	$1 \leq i, j \leq n+1, A \in N_G, \alpha\beta \in N_G^*$ the Φ_A constraint can be true
Axioms	$[i, i+1, \sigma^L_{ii+1}, B_i \rightarrow \bullet]$	$1 \leq i \leq n, B_i \in N_G, B_i \rightarrow \sigma_i \in P_G$ $\sigma_i \in T_G(\emptyset)$
Goals	$[i, j, \sigma^L_{ij}, A \rightarrow \alpha \Phi_A \bullet]$	$1 \leq i, j \leq n+1, A \in N_G,$
Inference Rules		
Prediction	$\frac{[i,j,\sigma^L_{ij}, B \rightarrow \beta \Phi_B \bullet]}{[i,i,\sigma^R_{ii}, A \rightarrow \bullet B \gamma \Phi_A]} \quad \langle A \rightarrow B\gamma \colon \Phi_A \rangle$	
Completion	$\frac{[i,j,\sigma^R_{ij}, A \rightarrow \alpha \bullet B\ \gamma\ \Phi_A] \quad [j,k,\sigma^L_{jk}, B \rightarrow \beta\ \Phi_B\ \bullet]}{[i,k,\sigma^R_{ik}, A \rightarrow \alpha\ B \bullet \gamma\ \Phi_A]}$	
Constraint	$\frac{[i,j,\sigma^R_{ij}, A \rightarrow \alpha \bullet \Phi_A]}{[i,j,\sigma^L_{ij}, A \rightarrow \alpha \Phi_A \bullet]} \quad \langle \vdash_{SLD} \Phi_A \rangle$	Φ_A is satisfiable

Table 3.1: The robust parsing deductive definition

$h^L_{ij} = \varphi(h^R_{ij})$. The substitution ν and the function φ are implicitly contained in the grammar constraint $\Phi_A(h^L_{ij}, h^R_{ij})$ performed as $\vdash_{SLD} \Phi_A$ (more details are presented in Chapter 4).

Table 3.1 presents the robust bottom-up active chart parser as a deductive system. In addition to the deductive system in (Shieber, Schabes, and Pereira, 1995), we have that: *each item is augmented with a syntagma*; and *the constraint rule* is a new inference rule. An inactive edge is obtained only after the constraint rule is applied, as the last inference rule. This corresponds to reaching a goal formula. Another specific aspect of our parsing mechanism is that it is robust, i.e., the goal items are associated to every nonterminal in the grammar, not only to the start symbol.

Definition 12 (Robust parsing provability). *Robust parsing provability corre-*

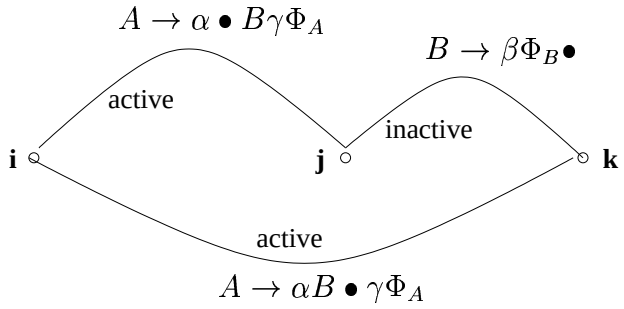


Figure 3.4: Chart parsing: completion rule

sponds to reaching the goal item:

$$\vdash_{rp} A(\sigma_{ij}^L) \textit{ iff } [i, j, \sigma_{ij}^L, A \rightarrow \alpha\Phi_A \bullet].$$

Thus, we can notice that the ground syntagma derivation used for defining the LWFG semantics is equivalent to robust parsing provability, i.e., $A \overset{*G}{\Rightarrow} \sigma$ iff $G \vdash_{rp} A(\sigma)$. The robust parsing algorithm is SPACE $O(n^2)$ and TIME O(n^3). The termination of the parser/generator will be discussed in Chapter 5 when we talk about LWFG decidability.

3.6 Representative Examples of a LWFG

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 (Brazma, 1993; Freivalds, Kinber, and Wiehagen, 1993; Lange, Nessel, and Wiehage, 1998). In this section, we formally define the representative examples of a Lexicalized Well-Founded Grammar. The representative examples will be used by our grammar learning model for the LWFG induction.

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

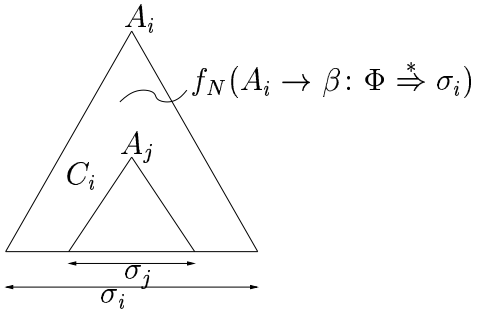


Figure 3.5: Syntagma partial ordering: $\sigma_i \succeq \sigma_j$.

introduce the notion of syntagma equivalence classes. The equivalence classes of syntagmas, eq_class(σ), are named by 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 $f_N(r \stackrel{*}{\Rightarrow} \sigma)$ the set of nonterminals that belong to the parse tree of $r \stackrel{*}{\Rightarrow} \sigma$, where $f_N\colon P_G \times E_\sigma \rightarrow 2^{N_G}, r \in P_G, \sigma \in E_\sigma$ (see Figure 3.5).

Definition 13. *A LWFG, G, is unambiguous if $\forall \sigma \in L_\sigma(G)$ there is one and only one rule $A \rightarrow \beta\colon \Phi \stackrel{*}{\Rightarrow} \sigma$.*[16]

The equivalence classes of a sublanguage E_σ of an unambiguous LWFG, G, are computed by Algorithm 2. For each syntagma $\sigma_i \in E_\sigma$ we choose the nonterminal A_i s.t. $A_i \rightarrow \beta\colon \Phi \stackrel{*}{\Rightarrow} \sigma_i$. The equivalence class of each subsyntagma $\sigma_j \subset \sigma_i$ is computed. Let C_i be the biggest nonterminal of the parsing subtree rooted at A_i, except the root A_i (Figure 3.5). 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,

[16]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 (see Section 5.1.1 — A1 and Definition 18).

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.

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. $A_i \rightarrow \beta\colon \Phi \stackrel{*}{\Rightarrow} \sigma_i$ /*given by the robust parser */
$i_\sigma \leftarrow gdl(\sigma_i)$ /*given by the robust parser */
$C_i \leftarrow$ Max_Nonterminal($f_N(r \stackrel{*}{\Rightarrow} \sigma_i) - \{A_i\}$) /*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)

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,$\succeq$, and the syntagma's ground derivation length, $gdl(\sigma_i)$:[17] $\sigma_m \geq$

[17]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.

$... \geq \sigma_i \geq ... \geq \sigma_0$. The algorithm is polynomial in $|E_\sigma|$ and $|\sigma|$. The procedure *eq_class* is efficiently performed by our robust parser.

Lemma 2. *Algorithm 2 ensures that any syntagma σ_i generated by an unambiguous LWFG, G, has the equivalence class greater than or equal to 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 14. *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 15. *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 $P_{G^-} \subset P_G$, we have that $E_\sigma \subseteq L_\sigma(G) \wedge E_\sigma \nsubseteq L_\sigma(G^-)$. 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.

We define a *representative sublanguage* E_σ of a grammar G to be any set of syntagmas which includes the representative examples, $E_R \subseteq E_\sigma \subseteq L_\sigma(G)$.

Given a representative sublanguage E_σ of an unambiguous LWFG G, such that $E_R^G \subseteq E_\sigma \subseteq L_\sigma(G)$, Algorithm 4[18] computes iteratively the totally ordered

[18] 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.

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} ensures 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 $r \leftarrow (A \rightarrow \beta : \Phi) \in P_G$ s.t. $A \rightarrow \beta : \Phi \stackrel{*G}{\Rightarrow} \sigma$ /*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 representative 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 $(G, E_\sigma) \stackrel{4}{\longrightarrow} (E_R^{G_r}, G_r)$.*

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. $r_i \overset{*G_r}{\Rightarrow} \sigma_i$ 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 $\sigma_i \in L_\sigma(G_r)$)

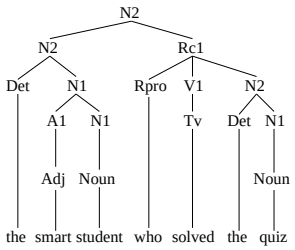
(iv) $r \notin P_{Gr}$ (We assume the contrary. By (iii), if $r \in P_{G_r}$, it follows that $r \overset{*G_r}{\Rightarrow} \sigma$. This implies that $\sigma \in L_\sigma(G_r)$ which contradicts (ii))

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 $r \overset{*G}{\Rightarrow} \sigma$. 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 $r \overset{*G_r}{\Rightarrow} \sigma$ (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 representative sublanguage E_σ complete w.r.t. an unambiguous LWFG G, 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-foundedness 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.

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



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

Example. Figure 3.6(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 3.1(a). Figure 3.6(a) shows the corresponding parse tree. For simplicity, we show only the strings without their semantic molecules.

Another example for illustrating Algorithm 2 and Algorithm 4 is given in Appendix A.6, where we show the generation of the representative examples and the equivalence classes for finite auxiliary verbs (the input is a sublanguage and the grammar for finite auxiliaries).

Chapter 4

Semantic Composition and Semantic Interpretation as Grammar Constraints

In the constraint-based grammars that are defined in this thesis, the semantic structures are composed by constraint solving, rather then functional application (with lambda expressions and lambda reduction). Moreover, the semantic interpretation is also encoded as a constraint, providing access to meaning during parsing.

The main goal of defining Lexicalized Well-Founded Grammars is to provide a formalism for natural language grammars such they are learnable from examples. Thus, the definition of semantic molecules, syntagmas and constraints must take the learning requirement into account. Learning from examples corresponds actually to learning the ground derivation step:

$$\frac{X \stackrel{*}{\Rightarrow} \sigma \qquad B_i \stackrel{*}{\Rightarrow} \sigma_i,\ i=1,\ldots,n, \sigma=\sigma_1 \circ \cdots \circ \sigma_n}{X(\sigma) \rightarrow B_1(\sigma_1),\ldots,B_n(\sigma_n) : \Phi(\bar{\sigma}),\ \bar{\sigma}=(\sigma,\sigma_1,\ldots,\sigma_n)}$$

In other words, if a positive example (σ) is not ground derived from the grammar, but its corresponding chunks (σ_i) can be derived, then learning the constraint rule will allow us to ground derive σ. In order to learn the rule, the nonterminal X

must be determined and the constraints $\Phi(\bar{\sigma})$ must be determined if $\sigma, \sigma_1, \ldots, \sigma_n$ are known.

This chapter presents the principles and properties needed to fulfill this learning requirement. In Section 4.1 we present the semantic composition constraints, while in Section 4.2 we describe the ontology-based semantic interpretation constraint.

4.1 Semantic Composition: Principles and Properties

The information for semantic composition is encoded in the head of the semantic molecules. There are three types of attributes that belong to the semantic molecule head: category attributes $\mathcal{A}_h^c$, variable attributes $\mathcal{A}_h^v$, and feature attributes $\mathcal{A}_h^f$. Thus $\mathcal{A}_h = \mathcal{A}_h^c \cup \mathcal{A}_h^v \cup \mathcal{A}_h^f$ and $\mathcal{A}_h^c, \mathcal{A}_h^v, \mathcal{A}_h^f$ are pairwise disjoint. For the semantic molecule $w' = \binom{h}{b}$ we have that $h = h^c \cup h^v \cup h^f$ and $var(h) = var(h^v) \subseteq var(b)$. For example, consider the semantic molecule for the noun phrase *major damage*.

$$\left(\begin{array}{l} \mathsf{h}\left[\begin{array}{ll} \mathsf{cat} & \mathsf{n1} \\ \mathsf{def} & \mathsf{no} \\ \mathsf{nr} & \mathsf{pl} \\ \mathsf{pers} & \mathsf{3} \\ \mathsf{case} & \mathsf{nomacc} \\ \mathsf{count} & \mathsf{y} \\ \mathsf{head} & X_{N1} \end{array}\right] \\ \mathsf{b}\left\langle X_{a1}.\mathsf{isa} = \mathsf{major},\ X_{N1}.X_{a3}{=}X_{a1},\ X_{N1}.\mathsf{isa}{=}\mathsf{damage} \right\rangle \end{array} \right)$$

We have that cat is a category attribute ($\in \mathcal{A}_h^c$), det, nr, pers, case, count are feature attributes ($\in \mathcal{A}_h^f$), while head is a variable attribute ($\in \mathcal{A}_h^v$).

We describe in turn each of these types of attributes and their corresponding principles. All principles, except the first and the last mirror principles in other constraint-based linguistic formalisms, such as HPSG (Pollard and Sag, 1994).

The category attributes $\mathcal{A}_h^c$ are state attributes, and their value set gives the category of the semantic molecule. There is one attribute, $\mathsf{cat} \in \mathcal{A}_h^c$, which is mandatory and whose value is the name of the category (e.g., $h.cat = adj, h.cat = noun$). The category of the semantic molecule can be given by: 1) the cat attribute alone, 2) the cat attribute together with other state attributes in $\mathcal{A}_h^c$ which are syntactic-semantic marks, or 3) the cat attribute together with only one attribute *bar*, which has integer values, as in X-bar theory (Chomsky, 1970; Jackendoff, 1977).

Principle 1 (Category Name Principle). *The category name h.cat of a generalized syntagma* $\sigma = (w, \binom{h}{b})$ *is the same as the grammar nonterminal augmented with the generalized syntagma* σ.

The above principle allows us to determine the name of the nonterminal in the left-hand side of the rule (X). For the example *major damage* the name of the nonterminal will be N ($h.cat = n$). In ILP terminology this is called *predicate name invention.*

The variable attributes $\mathcal{A}_h^v$ are attributes whose values are logical variables, which appear in the body of the semantic molecule as well (see Figure 4.1, page 83). Since the body, b, of the semantic molecule, w', is the semantic representation of the string w, the logical variables which are shared with the head h of the semantic molecule, represent the only semantic entities through which the semantic molecule w' can be bound to other semantic molecules. Thus, $\mathcal{A}_h^v$ is the semantic valence of the molecule, which allows the binding of the semantic representations,[1] giving the following principle:

Principle 2 (Semantic Representation Binding Principle). *All the logical variables which the body b of a semantic molecule shares with other semantic*

[1] Unlike $\mathcal{A}_h^v$, $\mathcal{A}_h^c$ allows the binding of the semantic molecules to the grammar nonterminals, and can be seen mainly as the syntactic valence of the molecules, which include the category.

molecules, are at the same time values of the variable attributes ($\mathcal{A}_h^v$) of the molecule head.

There is one variable attribute, $head \in \mathcal{A}_h^v$ that represents the semantic head of the semantic molecule (similar with the head notion in X-bar theory,[2] properties (iii), (iv)). Thus we can state the Semantic Head Principle:

Principle 3 (Semantic Head Principle). *For the composition of generalized syntagma , $\sigma = (w, \binom{h}{b})$, realized at the level of each grammar rule, there exists in the rule right-hand side, one and only one syntagma $\sigma_i = (w_i, \binom{h_i}{b_i})$, which has the same value of the attribute head, i.e., $h.head = h_i.head$.*

The feature attributes $\mathcal{A}_h^f$ are the attributes whose values express the specific properties of the semantic molecules (e.g., number, person).

Principle 4 (Feature Inheritance Principle). *All the feature attributes of a molecule head inherit the values of the corresponding attributes that belong to the semantic head (see Semantic Head Principle). That is, if h_i is the semantic head of h, $(h.head = h_i.head)$, then $h.f = h_i.f$, $\forall f \in \mathcal{A}_h^f \cap A_{h_i}^f$.*

Besides this principle, the feature attributes are used for category agreement. The categories that enter in agreement are maximum projection categories, if we are to use X-bar theory terminology. In general there are two types of agreement (Anderson, 2004): Modifier-Head Agreement (agreement between modifiers and the heads of their phrases) and Predicate-Argument Agreement (agreement between predicates, including verb, predicative adjective, and their arguments). The category that gives the agreement is known as the "controller" or "trigger", while the

[2] "The X-bar scheme is a restrictive mechanism for delimiting possible syntactic (or morphological) structures. General assumptions: (i) every X^n is a projection of X; (ii) X^{max} is the maximal projection of X; (iii) every phrase has a head determining its specific properties; (iv) the head properties are preserved in all projections; (v) a head category X combines with a non-head category Y which can be a complement (bar-level promotion: $X^n \rightarrow Y^{max}\ X^{n-1}$), an adjunct (bar-level preservation: $X^n \rightarrow Y^{max}\ X^n$), or a specifier (special case of bar-level promotion: $X^{max} \rightarrow Y^{max}\ X^{max-1}$)" (Avgustinova and Uszkoreit, 2001).

Type	**Controller/Trigger**	**Target**	**Features**
Modifier-Head	Noun	Adj	number gender
	Noun	Determiner	number person *gender (Rom)*
Predicate-Argument	Subject	Verb	number person gender
	Subject	Predicative Adjective	number gender

Table 4.1: Examples of agreement

category whose form is determined by agreement is called the "target". In Table 4.1 we present examples of agreement usually studied in linguistics (Anderson, 2004; Sag and Wasow, 1999) , which involve strict identities of features values between the trigger and the target.

However, the description of agreement should also take into account the coordination construction, which involves resolution rules (cf. Corbett as summarized in (Anderson, 2004)):

- Conjoined singulars typically yield a plural.
- Predominance of first over second over third person inside a coordination construction (i.e., he+I=we, not they).
- Gender resolution — e.g., in Romanian masc + fem = masc.

This linguistic knowledge about agreement is used in Section 4.1.2 in the form of the following principle:

Principle 5 (Feature Agreement Principle). *The agreeing/non-agreeing categories*[3] *and the agreement/non-agreement features are a-priori given based on linguistic knowledge, and are applied only at the semantic head level. They are language specific.*

Given all the above principles, we can now formulate the general Composition Principle:

Principle 6 (Composition Principle). *The semantic molecule corresponding to a rule left-hand side is obtained by the composition of semantic molecules corresponding to the nonterminals from the rule right-hand side, as in* (4.1).

$$
\begin{aligned}
w' = \begin{pmatrix} h \\ b \end{pmatrix} &= (w_1 \cdots w_n)' = w_1' \circ \cdots \circ w_n' \\
&= \begin{pmatrix} h_1 \\ b_1 \end{pmatrix} \circ \cdots \circ \begin{pmatrix} h_n \\ b_n \end{pmatrix} \\
&= \begin{pmatrix} h_1 \circ \cdots \circ h_n \\ \langle b_1, \ldots, b_n \rangle \nu \end{pmatrix}
\end{aligned} \tag{4.1}
$$

The composition of the molecule heads is realized by a set of constraints denoted by $\Phi_{comp}(h, h_1, ..., h_n) = (h \cup \bigcup_{i=1}^{n} h_i)\nu\mu$, where ν is the body variable specialization substitution and μ is head feature generalization substitution. The body parts are connected through conjunction after the application of the variable substitution ν, $b = (b_1, \ldots, b_n)\nu$. In the remainder of this thesis when we refer to compositional constraint we mean just the molecule head composition, Φ_{comp}.

In the next sections we present the composition properties regarding the substitutions ν (Section 4.1.1) and μ (Section 4.1.2), the compositional constraint, Φ_{comp} (Section 4.1.3) and its use in the generalization of the grammar rules (Section 4.1.4).

[3]Note that not all the categories have an agreeing/non-agreeing relation among them.

4.1.1 Body Variable Specialization Substitution(ν)

The body variable specialization substitution ν is the most general unifier (mgu) of b and $b_1, \ldots, b_n$, s.t $b = (b_1, \ldots, b_n)\nu = b_1\nu_1, \ldots, b_n\nu_n$. It is a particular form of the commonly used substitution (Lloyd, 2003), i.e., a finite set of the form $\{X_1/Y_1, \ldots, X_m/Y_m\}$, where $X_1, \ldots, X_m, Y_1, \ldots, Y_m$ are variables, and $X_1, \ldots, X_m$ are distinct. Each element X_i/Y_i is called a binding. If $\nu = \{X_1/Y_1, \ldots, X_m/Y_m\}$ then $domain(\nu) = \{X_1, \ldots, X_m\}$ and $range(\nu) = \{Y_1, \ldots, Y_m\}$. More exactly we have that:

$$
\begin{aligned}
&\nu = \{X_1/Y_1, \ldots, X_m/Y_m\} = \bigcup_{i=1}^{n} \nu_i \\
&domain(\nu) = \{X_1, \ldots, X_m\} = \bigcup_{i=1}^{n} domain(\nu_i) \\
&range(\nu) = \{Y_1, \ldots, Y_m\} \supseteq var(h) \\
&domain(\nu_i) \subseteq var(h_i)
\end{aligned}
\tag{4.2}
$$

where ν_i are pairwise disjoint, i.e., $\nu_i \cap \nu_j = \emptyset, 1 \leq i, j, \leq n$.

We will use the following notation for the body variable substitution:

$$\nu(\overline{X/Y}) = \{X/Y : X \in domain(\nu) \wedge Y \in range(\nu)\}$$

The body variable substitution composition $\nu_1\nu_2$ guarantees that $(domain(\nu_2) \cup range(\nu_2)) \cap domain(\nu_1) = \emptyset$, in order to avoid cycles. Thus, the composition is given by:

$$\nu_1\nu_2 = \nu_1(\overline{X/Y\nu_2}) \cup \nu_{2/1} \tag{4.3}$$

where $\nu_{2/1} = \{X/Y : (X \in domain(\nu_2) - range(\nu_1)) \wedge (Y \in range(\nu_2))\}$, represents the substitution ν_2 from which the elements whose domains were in the range of

ν_1 were removed. For example, in Figure 4.1 (page 83), given the substitution ν_{N1} and ν_{N2}, we have $\nu_{N2/N1} = \{X_{d1}/X_{N2}\}$, which explains why the term X_{N1}/X_{N2} is crossed-out from the composed substitution $\nu_{N2(N1)}$.

Property 1 (Body Variable Substitution). The body variable substitution ν is fully determined by the representative example,[4] as mgu of b and $b_1, \ldots, b_n$.

4.1.2 Head Feature Generalization Substitution (μ)

For the feature attributes, a contextual constant generalization substitution, μ, is used. The μ substitution is used for generalization of the feature attribute values needed for agreement constraints. Extending the notation, we will assign the index 0 to the head h corresponding to the left-hand side nonterminal (i.e., $h = h_0$).

$$
\begin{aligned}
&\mu = \{t_1/X_1 \colon e_1, \ldots, t_m/X_m \colon e_m\} \\
&domain(\mu) = \{t_1, \ldots, t_m\} \\
&range(\mu) = \{X_1, \ldots, X_m\} \\
&environment(\mu) = \{e_1, \ldots, e_m\} \\
&range(\mu) \cap (domain(\nu) \cup range(\nu)) = \emptyset \\
&\mu = \bigcup_{i=0}^{n} \mu_i
\end{aligned}
\tag{4.4}
$$

where ν is the body variable substitution, and $environment(\mu_i)$ are pairwise disjoint, i.e., $environment(\mu_i) \cap environment(\mu_j) = \emptyset, 0 \leq i, j, \leq n$. The $environment(\mu_i)$, $0 \leq i \leq n$, are feature attributes of the ith component (e.g., $\{h_i.f_1, h_i.f_2, \ldots\}$), while $domain(\mu_i)$ are the values of the feature attributes. We denote $e_j = e(t_j/X_j)$.

[4] We remind our reader that representative examples are used to learn the grammar rules and the compositional constraints.

The composition $\mu_1\mu_2$ of the contextual constant generalization substitution is:

$$\begin{aligned}\mu_1\mu_2 = \{s\colon \textbf{if} \quad & (t_1/X : e_1) \in \mu_1 \ \wedge \ (t_2/Y : e_2) \in \mu_2 \ \wedge \ e_1 \cap e_2 \neq \emptyset \\ & \textbf{then} \quad s = (t_1/X\colon e_1 - e_2, t_2/Y : e_2 - e_1, X/Y) \\ & \textbf{else} \quad s = (t_1/X\colon e_1, t_2/Y\colon e_2)\}\end{aligned} \tag{4.5}$$

In this way, the variables X, Y, which generalize the constants of the same environment $e_1 \cap e_2$ of μ_1 and μ_2, become the same by the X/Y substitution.

Algorithm 5: Feature_Generalization$(h_0, h_1, \ldots, h_n)$

```
    μ ← ∅
    if h_sem.head = h.head, 1 ≤ sem ≤ n then
    |  /*h_sem is the Semantic Head. Principle 3                          */
1   |  foreach f ∈ A_h^f ∧ h.f = h_sem.f = t do
    |  |  /*Principle 4                                                   */
    |  |  μ ← μ ∪ {t/X : (h.f, h_sem.f)}
2   |  foreach j = 1..n ∧ h_sem.cat = c_sem ∧ h_j.cat = c_j ∧ agree(c_sem, c_j) do
    |  |  /*Basic treatment of Principle 5, without details               */
    |  |  foreach f ∈ agreeing_features(c_sem, c_j) do
    |  |  |  if h_sem.f = h_j.f = t then
    |  |  |  |  μ ← μ ∪ {t/X : (h_sem.f, h_j.f)}
    |  |  foreach f ∈ non_agreeing_features(c_sem, c_j) do
    |  |  |  if h_j.f = t then
    |  |  |  |  μ ← μ ∪ {t/X : (h_j.f)}
    return μ
```

Property 2 (Head Feature Substitution). The contextual constant generalization substitution μ is fully determined by the representative example and *a priori* linguistic knowledge given as agreement rules (Algorithm 5). The μ substitution is sound, even if it can be incomplete if the learned grammar rule is too specialized and thus it does not contain the categories given in the agreement rules, at the semantic head level.

4.1.3 Compositional Constraint Determinacy

The compositional semantic constraint is:

$$\Phi_{comp}(h, h_1, \ldots, h_n) = \bigcup_{i=0}^{n} h_i^c \ \cup \ \bigcup_{i=0}^{n} h_i^v \nu_i \ \cup \ \bigcup_{i=0}^{n} h_i^f \mu_i = (\bigcup_{i=0}^{n} h_i)\nu\mu \tag{4.6}$$

In the above formula, and herein, we have that $h = h_0, \nu_0 = \{\}$. The set of constraints, $\Phi_{comp}(h, h_1, ..., h_n)$, is encoded as a system of equations, (4.7), and will be learned together with the grammar rule during the induction process. This system of equations is similar to "path equations" (Shieber et al., 1983; van Noord, 1993), but is applied to flat feature structures (our semantic molecule heads).

$$\left\{ \begin{array}{l} h_i.c = constant \\ h_i.v_i = h_j.v_j \\ h_i.f = constant \quad or \quad h_i.f = h_j.f \end{array} \right\} \quad \text{where} \quad \begin{array}{l} 0 \leq i, j \leq n, i \neq j \\ c \in \mathcal{A}^c_{h_i} \\ v_i \in \mathcal{A}^v_{h_i}, v_j \in \mathcal{A}^v_{h_j} \\ f \in \mathcal{A}^f_{h_i}, f \in \mathcal{A}^f_{h_j} \end{array} \tag{4.7}$$

Since the ν, μ substitutions can be determined by example, then the above system of equations can be generated by Algorithm 6.

Property 3 (Compositional Constraint Determinacy). The compositional constraint Φ_{comp} is fully determined by example using Algorithm 6. Moreover, at the rule level, Φ_{comp} fully determines the output of the parser and generator.

Given a rule $A(\sigma) \rightarrow B_1(\sigma_1), \ldots, B_n(\sigma_n) : \Phi_{comp}(h, h_1, \ldots, h_n)$, using the definition of the ground syntagma derivation (Definition 6) in DCG-provability form we have:

$$\frac{\vdash B_i(\sigma_i), i{=}1..n \qquad \vdash \Phi_{comp}(h, h_1, ..., h_n)}{\vdash A(\sigma)}$$

Algorithm 6: Generate_Compositional_Constraints$(\sigma_0, \sigma_1, \ldots, \sigma_n)$

$\sigma_i = (w_i, \binom{h_i}{b_i}), 0 \leq i \leq n$
$\Phi_{comp} \leftarrow \emptyset$
$\nu \leftarrow mgu(b_0, (b_1, \ldots, b_n))$
$\mu \leftarrow$ Feature_Generalization$(h_0, h_1, \ldots, h_n)$
foreach $0 \leq i \leq n \wedge c \in \mathcal{A}^c_{h_i} \wedge h_i.c = t \in h^c_i$ **do**
  $\Phi_{comp} \leftarrow \Phi_{comp} \cup \{h_i.c = t\}$
foreach $X/Y \in \nu \wedge v_i \in \mathcal{A}^v_{h_i} \wedge h_i.v_i = X \wedge v_j \in \mathcal{A}^v_{h_j} \wedge h_j.v_j = Y$ **do**
  $\Phi_{comp} \leftarrow \Phi_{comp} \cup \{h_i.v_i = h_j.v_j\}$
foreach $0 \leq i \leq n \wedge f \in \mathcal{A}^f_{h_i} \wedge h_i.f = t \in h^f_i$ **do**
  if $(t/X : e(t/X) \notin \mu \vee h_i.f \notin e(t/X)$ **then**
    $\Phi_{comp} \leftarrow \Phi_{comp} \cup \{h_i.f = t\}$
  else if $\exists h_j.f \in e(t/X) \wedge i < j$ **then**
    $\Phi_{comp} \leftarrow \Phi_{comp} \cup \{h_i.f = h_j.f\}$
return Φ_{comp}

From Definition 9 and 10 of parsing and generation, it results that σ can be computed either by specifying the string w, or by specifying its semantic representation b. In both cases, we have that:

$$
\begin{aligned}
& w = w_1, \ldots, w_n \\
& b = (b_1, \ldots, b_n)\nu \quad (Principles\ \ 6\ \ and\ \ 2) \\
& \Phi_{comp}(h, h_1, \ldots, h_n) = (h \cup \bigcup_{i=1}^{n} h_i)\nu\mu \quad (Principles\ \ 6\ \ and\ \ 3,\ \ 4,\ \ 5) \\
& h \stackrel{\vdash\Phi}{=} (\bigcup_{i=1}^{n} h_i)\nu\mu
\end{aligned}
\tag{4.8}
$$

The last equation in (4.8) gives the value of h as a consequence of compositional constraint proving, $\vdash \Phi$ (see Section 3.5.1.2, Table 3.1 page 61).

4.1.4 Rule Generalization

In this section we give a rule generalization property (Property 4, page 84), which will be used in Chapter 5, Section 5.3.1 for the foundation of the search space of the LWFG induction. For this, we first define the properties of the rule derivation step (Section 4.1.4.1), and then we give the properties of the rule generalization step (Section 4.1.4.2).

4.1.4.1 Rule Derivation Step

If $\beta = B_1(\sigma_1), \ldots, B_k(\sigma_k)$ and $\sigma_i = (w_i, \binom{h_i}{b_i})$, for $1 \leq i \leq k$, below we use the following notation: $h_\beta = \bigcup_{B_i(\sigma_i) \in \beta} h_i$, $b_\beta = \bigcup_{B_i(\sigma_i) \in \beta} b_i$, and for provability we use $\vdash \beta$ for $\vdash B_i(\sigma_i), 1 \leq i \leq k$. Thus for the rules:

$$\begin{aligned} &B(\sigma_B) \rightarrow \beta \colon \Phi_B \\ &A(\sigma_A) \rightarrow \alpha B(\sigma_B^*)\gamma \colon \Phi_A \end{aligned} \tag{4.9}$$

we have (given (4.8)):

$$\begin{aligned} b_B &= b_\beta \nu_B \\ \Phi_B &= (h_B \cup h_\beta)\nu_B \mu_B \\ h_B &\stackrel{\vdash \Phi_B}{=} h_\beta \nu_B \mu_B \\ b_A &= b_\alpha b_B^* b_\gamma \nu_A \\ \Phi_A &= (h_A \cup h_\alpha \cup h_B^* \cup h_\gamma)\nu_A \mu_A \\ h_A &\stackrel{\vdash \Phi_A}{=} (h_\alpha \cup h_B^* \cup h_\gamma)\nu_A \mu_A \end{aligned} \tag{4.10}$$

Given the above two rules, we consider the rule derivation step:

$$\frac{A(\sigma_A)\to\alpha B(\sigma_B^*)\gamma\colon \Phi_A \qquad B(\sigma_B)\to\beta\colon \Phi_B}{A(\sigma_A)\to\alpha\beta\gamma\colon \Phi_{A(B)}} \tag{4.11}$$

Substituting B we have: $A(\sigma_A) \to \alpha(\beta\colon \Phi_B, \sigma_B = \sigma_B^*)\gamma\colon \Phi_A$, and the equivalent DCG-provability form of the rule derivation step is:

$$\dfrac{\dfrac{\dfrac{\vdash\beta \quad \vdash\Phi_B}{\vdash B(\sigma_B)} \qquad \dfrac{\vdash\alpha B(\sigma_B^*)\gamma \quad \vdash\Phi_A}{\vdash A(\sigma_A)}}{\vdash\alpha\beta\gamma \quad \vdash\Phi_B,\sigma_B=\sigma_B^*,\Phi_A}}{\vdash A(\sigma_A)} \tag{4.12}$$

If we denote $\Phi_{A(B)} = (\Phi_B, \sigma_B = \sigma_B^*, \Phi_A)$ we have for the derived rule:

$$\begin{aligned} A(\sigma_A) &\to \alpha\beta\gamma\colon \Phi_{A(B)} \\ b_A &= b_\alpha b_\beta b_\gamma \nu_{A(B)} \\ \Phi_{A(B)} &= (h_A \cup h_\alpha \cup h_\beta \cup h_\gamma)\nu_{A(B)}\mu_{A(B)} \\ h_A &\overset{\vdash\Phi_{A(B)}}{=} (h_\alpha \cup h_\beta \cup h_\gamma)\nu_{A(B)}\mu_{A(B)} \end{aligned} \tag{4.13}$$

Lemma 3 (Rule Derivation Step). *Given the two rules in* (4.9) *and the derived rule in* (4.13)*, the substitutions of the derived rule are obtained by composing the substitutions of the initial rules:*

$$\begin{aligned} \nu_{A(B)} &= \nu_B\nu_A \\ \mu_{A(B)} &= \mu_B\mu_A \end{aligned} \tag{4.14}$$

Proof. Since $b_B = b_B^*$ and $h_B = h_B^*$, from (4.10) we have that:

$$b_A = b_\alpha b_\beta \nu_B b_\gamma \nu_A = b_\alpha b_\beta b_\gamma \nu_B \nu_A \tag{4.15}$$

and from (4.13) results $\nu_{A(B)} = \nu_B\nu_A$.

Similarly from (4.10) we have that $h_A \overset{\vdash\Phi_{A(B)}}{=} (h_\alpha \cup h_\beta\nu_B\mu_B \cup h_\gamma)\nu_A\mu_A \overset{\vdash\Phi_{A(B)}}{=} (h_\alpha \cup h_\beta \cup h_\gamma)\nu_B\nu_A\mu_B\mu_A$, and thus from (4.13) and (4.15) it results that $\mu_{A(B)} = \mu_B\mu_A$.

□

4.1.4.2 Rule Generalization Step

The rule generalization step is the inverse of the rule derivation step. Let the specialized rule (4.16):

$$
\begin{aligned}
A(\sigma_A) &\rightarrow \alpha\beta\gamma\colon \Phi'_A \\
b_A &= b_\alpha b_\beta b_\gamma \nu'_A \\
\Phi'_A &= (h_A \cup h_\alpha \cup h_\beta \cup h_\gamma)\mu'_A \\
h_A &\overset{\vdash \Phi'_A}{=} (h_\alpha \cup h_\beta \cup h_\gamma)\nu'_A \mu'_A
\end{aligned}
\tag{4.16}
$$

and let the rule generalization step (4.17):

$$
\frac{A(\sigma_A)\rightarrow\alpha\beta\gamma\colon \Phi'_A \qquad B(\sigma_B)\rightarrow\beta\colon \Phi_B}{A(\sigma_A)\rightarrow\alpha B(\sigma^*_B)\gamma\colon \Phi_A}
\tag{4.17}
$$

The rule $A(\sigma_A) \rightarrow \alpha\beta\gamma\colon \Phi_{A(B)}$ (4.13) derived from the generalized rule (4.9), which are the same as the rules in the generalization step, subsumes the specialized rule $A(\sigma_A) \rightarrow \alpha\beta\gamma\colon \Phi'_A$ (4.16):

$$
\begin{aligned}
\nu'_A &= \nu_{A(B)} \\
\mu'_A &\subseteqq \mu_{A(B)}
\end{aligned}
\tag{4.18}
$$

The $\subseteqq$ relation present in the equation (4.18) is explained by Property 2, which states that the contextual constant generalization is always applied to the semantic head projection (step 1 in Algorithm 5), while the agreement is performed only at the semantic head level (step 2 in Algorithm 5). For example, if the semantic head of the rule in (4.16) is contained in α, then no constant generalization in β is possible in the context of this specialized rule. However, in the first rule of (4.9) the semantic head is in β and thus the generalization is performed.

[5] We use the equivalence relation because the substitutions can differ by the constant values which are generalized (see Figure 4.1, for μ'_{N2} and $\mu_{N2(N1)}$).

I. Semantic Molecules

I-1 $(major/adj)' = h_{adj} \bowtie b_{adj} = \begin{bmatrix} cat & adj \\ head & X_{a1} \\ mod & X_{a2} \end{bmatrix} \bowtie \langle X_{a1}.isa = major, X_{a2}.X_{a3} = X_{a1} \rangle$

I-2 $(damage/noun)' = h_{noun} \bowtie b_{noun} = \begin{bmatrix} cat & noun \\ nr & sg \\ head & X_{n1} \end{bmatrix} \bowtie \langle X_{n1}.isa = damage \rangle$

I-2' $(damages/noun)' = h_{noun} \bowtie b_{noun} = \begin{bmatrix} cat & noun \\ nr & pl \\ head & X_{n1} \end{bmatrix} \bowtie \langle X_{n1}.isa = damage \rangle$

I-3 $(a/det)' = h_{det} \bowtie b_{det} = \begin{bmatrix} cat & det \\ nr & sg \\ mod & X_{d1} \end{bmatrix} \bowtie \langle X_{d1}.det = a \rangle$

I-4 $(major \;\; damages)' = h_{N1} \bowtie b_{N1}$

$$= \begin{bmatrix} cat & n1 \\ nr & pl \\ head & X_{N1} \end{bmatrix} \bowtie \langle X_{a1}.isa = major, X_{N1}.X_{a3} = X_{a1}, X_{N1}.isa = damage \rangle$$

I-5 $(a \;\; major \;\; damage)' = h_{N2} \bowtie b_{N2}$

$$= \begin{bmatrix} cat & n2 \\ nr & sg \\ head & X_{N2} \end{bmatrix} \bowtie \langle X_{N2}.det = a, X_{a1}.isa = major, X_{N2}.X_{a3} = X_{a1}, X_{N2}.isa = damage \rangle$$

II. Constraint Grammar Rules

II-1 $\mathbf{N1}(\sigma_{\mathbf{10}}) \rightarrow Adj(\sigma_{11}),\ Noun(\sigma_{12}) : \Phi_{N1}$

$\Phi_{N1} = (h_{10} \cup h_{11} \cup h_{12}) \nu_{N1} \mu_{N1}$

$\nu_{N1} = \{X_{n1}/X_{N1}, X_{a2}/X_{N1}\}$

$\mu_{N1} = \{pl/Y_1 : (\mathbf{h_{10}.nr}, h_{12}.nr)\}$

$\sigma_{21} = \sigma_{31} = \sigma_{det}$

$\sigma_{\mathbf{10}} = \sigma_{\mathbf{22}} = \sigma_{N1}$

$\sigma_{11} = \sigma_{32} = \sigma_{adj}$

$\sigma_{12} = \sigma_{33} = \sigma_{noun}$

$\sigma_{20} = \sigma_{30} = \sigma_{N2}$

II-2 $N2(\sigma_{20}) \rightarrow Det(\sigma_{21}),\ \mathbf{N1}(\sigma_{\mathbf{22}}) : \Phi_{N2}$ /* generalized rule*/

$\Phi_{N2} = (h_{20} \cup h_{21} \cup h_{22}) \nu_{N2} \mu_{N2}$

$\nu_{N2} = \{X_{N1}/X_{N2}, X_{d1}/X_{N2}\}$

$\mu_{N2} = \{sg/Y_2 : (h_{20}.nr, h_{21}.nr, \mathbf{h_{22}.nr})\}$

II-1+2 $N2(\sigma_{20}) \rightarrow Det(\sigma_{21}),\ Adj(\sigma_{11}),\ Noun(\sigma_{12}) : \Phi_{N2(N1)}$ /* derived rule */

$\Phi_{N2(N1)} = (h_{20} \cup h_{21} \cup h_{11} \cup h_{12}) \nu_{N2(N1)} \mu_{N2(N1)}$

$\nu_{N2(N1)} = \{X_{n1}/X_{N2}, X_{a2}/X_{N2},\ X_{N1}/X_{N2},\ X_{d1}/X_{N2}\}$

$\mu_{N2(N1)} = \{pl/Y_1 : h_{12}.nr, sg/Y_2 : (h_{20}.nr, h_{21}.nr), Y_1/Y_2\}$

II-3 $N2(\sigma_{30}) \rightarrow Det(\sigma_{31}),\ Adj(\sigma_{32}),\ Noun(\sigma_{33}) : \Phi'_{N2}$ /* specialized rule */

$\Phi'_{N2} = (h_{30} \cup h_{31} \cup h_{32} \cup h_{33}) \nu'_{N2} \mu'_{N2}$

$\nu'_{N2} = \{X_{d1}/X_{N2}, X_{a2}/X_{N2}, X_{n1}/X_{N2}\}$

$\mu'_{N2} = \{sg/Y : (h_{30}.nr, h_{31}.nr, h_{33}.nr)\}$

Figure 4.1: The Rule Derivation/Generalization Step

Property 4 (Rule Generalization). The rule generalization step (4.17) guarantees that $L_\sigma(A(\sigma_A) \rightarrow \alpha B(\sigma^*_B)\gamma\colon \Phi_A) \supseteq L_\sigma(A(\sigma_A) \rightarrow \alpha\beta\gamma\colon \Phi'_A)$.

The above property holds due to relation (4.18) and to the fact that the nonterminal B can have multiple definitions (thus, $\mu_{A(B)}$ and $\nu_{A(B)}$ have multiple values).[6]

4.1.5 Example

Figure 4.1 shows an example. I-1, ..., I-3 show the elementary semantic molecules of the adjective *major*, the nouns *damage* (sg) and *damages* (pl), as well as the determiner *a*. I-4 is the derived semantic molecule of the noun phrase *major damages* ground derived by the rule II-1, while I-5 is the derived semantic molecule of the noun phrase *a major damage*, which can be ground derived from either rules II-2, II-1+2, II-3.

From the constraint grammar rules II-1 and II-2 is obtained the derived rule II-1+2 by the rule derivation step with $\sigma_{22} = \sigma_{10}$:

$$\frac{N2(\sigma_{20}) \rightarrow Det(\sigma_{21}), N1(\sigma_{22})\colon \Phi_{N2} \qquad N1(\sigma_{10}) \rightarrow Adj(\sigma_{11}), Noun(\sigma_{12})\colon \Phi_{N1}}{N2(\sigma_{20}) \rightarrow Det(\sigma_{21}), Adj(\sigma_{11}), Noun(\sigma_{12})\colon \Phi_{N2(N1)}}$$

and from the rules II-3 and II-1 is obtained the generalized rule II-2 by the rule generalization step with $\sigma_{30} = \sigma_{20}$,$\sigma_{31} = \sigma_{21}$, $\sigma_{32} = \sigma_{11}$, $\sigma_{33} = \sigma_{12}$, and $\sigma_{10} = \sigma_{22}$:

$$\frac{N2(\sigma_{30}) \rightarrow Det(\sigma_{31}), Adj(\sigma_{32}), Noun(\sigma_{33})\colon \Phi'_{N2} \qquad N1(\sigma_{10}) \rightarrow Adj(\sigma_{11}), Noun(\sigma_{12})\colon \Phi_{N1}}{N2(\sigma_{20}) \rightarrow Det(\sigma_{21}), N1(\sigma_{22})\colon \Phi_{N2}}$$

For all the constraint rules involved in the derivation and generalization steps, the compositional constraint Φ together with the variable substitution ν

[6] Property 4 still holds even if the μ substitution is absent (i.e., we do not have contextual constant generalization and every rule is specific to particular values of feature attributes).

and the contextual constant generalization μ are shown in Figure 4.1. It can be noticed the variable substitution composition (4.3) $\nu_{N1}\nu_{N2} = \nu_{N2(N1)}$, as well as the composition of the contextual constant generalization substitution (4.5) $\mu_{N1}\mu_{N2} = \mu_{N2(N1)}$. We can also notice that the relation (4.18) holds. In this case we have that $\nu'_{N2} = \nu_{N2(N1)}$ and $\mu'_{N2} \equiv \mu_{N2(N1)}$.

4.2 Ontology-based Semantic Interpretation

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 afterwards during language analysis, and it is built using a meta-interpreter with *freeze* (Saraswat, 1989) (Muresan, Potolea, and Muresan, 1998). The meta-interpreter ensures that the atomic predicates, APs, (see Eq. (3.1)), 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. Moreover, 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 *major damages* (Figure 4.1) does not need to store the slot X_{a3}, a fact that allows us to use flat feature structures for representing the head of the semantic molecule. At this point, when Φ_{onto} is applied, the variable X_{a3} becomes instantiated with the value taken from the ontology (e.g., *degree*). The ontology-based semantic interpretation constraint is important for the dis-

ambiguation required for some phenomena (e.g., prepositional phrase attachment, coordinations), and for the semantic interpretation of phenomena not usually analyzed by current broad-coverage grammars or statistical syntactic parsers (e.g., prepositions, noun-noun compounds). More details are presented in Chapter 6.

Parsing Reversibility Principle. The string representation part of the semantic molecule (OntoSeR) guarantees the parsing reversibility preserving the original string meaning. Thus, the generalized syntagma is independent of the ontology-based interpretation level (e.g., interpretation by Φ_{onto}).

Uniform Representation Principle. The string representation part of the semantic molecule (OntoSeR) is independent of the knowledge level where the acquisition take place: terminological knowledge (ontology with general concepts), factual knowledge (knowledge base with individuals), and discourse (text knowledge). Thus, we consider that the assertional form K_σ of a syntagma σ is the same regardless of the asserting level (K_o, K_i, K_d) that is ontology, factual or discourse level, respectively. K_σ is dependent on the interpretation level given by Φ_{onto}. If $\sigma = (w, \binom{h}{b})$, b is guaranteed to preserve the whole meaning of w at the grammar level, while K_σ is dependent on Φ_{onto}, but independent of the knowledge level where the acquisition take place. Thus, the meta-interpreter which perform Φ_{onto} guarantees the interface with different KRS (knowledge representation systems).

Natural Language as Problem Formulation Principle. The discourse knowledge (assertional) representation K_d, is only the logic-based problem formulation that can be further solved using logic as problem solving (Kowalski, 1979). That is, the meta-interpreter Φ_{onto} does not deal with deep reasoning at the level of K_d assertion. In other words, we are concerned only with the meaning explicitly

given in text. Thus, K_d can contain the representation of a paradox formulation in natural language, even if the reasoning about its solution cannot be emphasized. The evaluation has to validate the correctness of problem formulation in natural language. This can be done by all possible queries (questions) relevant to a discourse, validated by human subjects at the string level (answers). As a summary of Chapter 3, the assertional form K_σ of a syntagma σ can be obtained by relation (4.19). More details will be presented in Chapter 6, specifically in Figure 6.2.

$$
\begin{aligned}
& A(\sigma) \to B_1(\sigma_1), \ldots, B_n(\sigma_n) \colon \Phi(\bar{\sigma}) \\
& \sigma = \sigma_1 \ldots \sigma_n \\
& \sigma = (w, \binom{h}{b}) = \sigma_0(w_0, h_0 \bowtie b_0) \\
& \sigma_i = (w_i, h_i \bowtie b_i), 1 \leq i \leq n \\
& w = w_1 \cdots w_n \\
& b = (b_1, \ldots, b_n)\nu \\
& \Phi = \Phi_{comp}(\sigma), \Phi_{onto}(b) \\
& \Phi_{comp}(\sigma) = (\bigcup_0^n h_i)\nu\mu \\
& h \overset{\vdash \Phi_{comp}}{=} (\bigcup_1^n h_i)\nu\mu \\
& K_\sigma = \Phi_{onto}(b)
\end{aligned}
\tag{4.19}
$$

In Section 3.3 and Section 3.4 we have defined the operational semantics of a grammar G as the set of all syntagmas ground derived (generated) by G. Due to Φ_{onto} constraint at the grammar rule level, it results that the semantics $L_\sigma(G)$ is determined not only by the grammar but also by the ontology, i.e., $L_\sigma(G) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, \exists A \in N_G, K \wedge P_G \vdash A(\sigma)\}$, where $K \supseteq K_o \cup K_i \cup K_d$. We

have:

$\sigma \in L_\sigma(G)$ iff $K_o, K_i, K_d \vdash K_\sigma$ ($\vdash$ means derived as acceptable).

The meaning, M_σ, of a syntagma, σ, is defined as the set of all query-syntagmas which become derivable by adding the assertional logical form K_σ to the system knowledge (K_o, K_i, K_d).

$$M_\sigma = \{\sigma_Q | (K_o, K_i, K_d \nvdash K_{\sigma_Q}) \wedge (K_o, K_i, K_d, K_\sigma \vdash K_{\sigma_Q})\} \tag{4.20}$$

where K_{σ_Q} includes the answer to the question as well.

Similarly, we can define the discourse meaning:

$$M_d = \{\sigma_Q | (K_o, K_i \nvdash K_{\sigma_Q}) \wedge (K_o, K_i, K_d \vdash K_{\sigma_Q})\} \tag{4.21}$$

Chapter 8 contains more discussions about meaning as answers to questions.

4.2.1 The Semantic Interpreter

Semantic interpretation is performed at the rule level through $\Phi_{onto}(b)$. Given the definition of OntoSeR in (3.1) and the notation $\Phi_{onto}(b) = b'$, the interpretation of OntoSeR is given below:.

$$(AP)' \leftarrow (postpone\ (AP))'$$

$$(OntoSeR_1\ \langle \text{lop} \rangle\ OntoSeR_2)' \leftarrow OntoSeR_1'\ \langle \text{lop} \rangle\ OntoSeR_2'$$

$$postpone\ (AP) \leftarrow freeze\ (X \in var\ (AP)\ , AP)$$

The above definition entails that an atomic predicate, AP, is postponed through the $freeze$ predicate until at least one of its variables becomes instantiated. This allows a nondeterministic efficient search in the ontology. The search strategy of the meta-interpreter is independent of the actual representation of the ontology, allowing an interface with any ontology at the level of atomic predicate meaning.

The meta-interpreter can be enhanced with generative ontology[7] axioms (Jensen and Nilsson, 2003): $X' \leftarrow X.isa = X'$, $(X.Y = Z)' \leftarrow X'.Y' = Z'$ (admissible concept rule), $Y = Z \leftarrow X.Y = X.Z$ (well-formedness principle for distinct simultaneous roles), $X.Y = Z \leftrightarrow Z.Y^{-1} = X$ (inversion principle), and also with a set of admissible affinities and role relations specified as atomic axioms. The latter refers to the ontologically admissible combinations of concepts and relations (e.g., $event.agt = substance$, $agt.isa = by$). In our current implementation of the semantic interpreter we only partially use these axioms (see Chapter 8).

The OntoSeR is an ontology independent semantic representation, in the same way an ontology is a language independent logical structure. Two predicates are implemented for asserting to and querying the ontology respectively. In the querying process, different OntoSeRs can have the same answer, thus transforming the problem of logical equivalence viewed as "meaning identity" (Shieber, 1994) into equivalence viewed as concept identity. This ensures the computational tractability requirement for our semantic framework. More discussion is given in Chapter 6 and Chapter 8. In Chapter 8 we define the weak concept identity principle for our semantic interpreter, related to terminological knowledge.

[7]Starting from a skeleton ontology, generative ontologies are formed by rules for combining concepts using semantic roles (binary relations) as binders: "The role relations express possible relations among the nodes in the lattice constituting the ontology. Thereby they make possible the generation of an infinite number of ontological nodes in the lattice, thus establishing a generative ontology. [...] The notion of generative ontology is inspired by the generative grammar paradigm and provides semantic domains for a compositional ontological semantics for NPs containing PPs. In contrast to traditional logical semantics, which strongly emphasizes the semantic contribution of determiners, our ontological semantics places decisive weight on the conceptual semantics of the nominal parts of NPs and their modifiers such as PPs." (Jensen and Nilsson, 2003).

Chapter 5

Induction of Lexicalized Well-Founded Grammars

In this chapter we describe our theoretical LWFG learning model. We begin by presenting the learning algorithms in Section 5.1. Our learning model allows grammar learning both from ordered representative examples (Section 5.1.2) and unordered representative examples (Section 5.1.3). Section 5.2 presents the way in which grammar merging is defined in our framework. In Section 5.3, we provide the foundation of the hypothesis search space as Boolean algebra/complete grammar lattice, together with the learnability theorem. This section concludes with the presentation of the LWFG induction in ILP setting and the description of our Grammar Approximation by Representative Sublanguage (GARS) model.

5.1 Grammar Learning

The grammar induction problem can be stated as follows: given a representative 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 $G = G'$. In other words, the learning should always converge. In order to

prove this convergence (see Theorem 2 pg. 102, Theorem 3 pg. 104, and Theorem 8 pg. 130), we need to formulate some properties and assumptions regarding our Lexicalized Well-Founded Grammars. The work presented in this section is based on our papers (Muresan, 2004; Muresan, Muresan, and Klavans, 2004; Muresan, Muresan, and Klavans, 2005)

5.1.1 Properties and Assumptions

A 1. A property of Lexicalized Well-Founded Grammars (see Definition 5, (vii), pg. 51) 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 $\sigma_1 = (w_1, \binom{h_1}{b_1})$ and $\sigma_2 = (w_2, \binom{h_2}{b_2})$ 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* $(\forall \sigma_1 \in L_{\sigma 1} \;\; \exists \sigma_2 \in L_{\sigma 2} \;\; s.t. \;\; \sigma_1 \equiv \sigma_2) \wedge (\forall \sigma_2 \in L_{\sigma 2} \;\; \exists \sigma_1 \in L_{\sigma 1} \;\; s.t. \;\; \sigma_1 \equiv \sigma_2)$

(iii) The intersection by equivalence of two sets of syntagmas $L_{\sigma 1}$ and $L_{\sigma 2}$ is the set $L_{\sigma 1} \bigcap_{\equiv} L_{\sigma 2} = \{\sigma_1 \equiv \sigma_2 | \sigma_1 \in L_{\sigma 1} \wedge \sigma_2 \in L_{\sigma 2}\}$.

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 3.4). 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 3.6, we consider only unambiguous LWFGs. Two points should be made:

(i) Unambiguity refers to syntagmas and not to natural language expressions (strings). Two syntagmas $\sigma_1 = (w_1, \binom{h_1}{b_1})$ and $\sigma_2 = (w_2, \binom{h_2}{b_2})$ 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 (more details are given in Section 6.4 and Appendix B). 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.

(ii) Unambiguity refers to syntagmas (i.e., representations) and not to interpreted natural language expressions. This means that a syntagma derived from a single rule can have many interpretations. Let us take the example (*bone knife*, $[cat\colon n, ...] \bowtie [X_1.isa = \text{bone}, X_2.Y = X_1, X_2.isa = \text{knife}]$). It 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 dissertation, we will not consider the interpretation ambiguity that is handled by the Φ_{onto} constraint, which nondeterministically returns all interpretations.

Definition 16. *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 $A \to \beta_i\colon \Phi_i \neq A \to \beta_j\colon \Phi_j$ iff $L_\sigma(A \to \beta_i\colon \Phi_i) \neq L_\sigma(A \to \beta_j\colon \Phi_j)$, respectively.*[1]

Lemma 4. *An unambiguous LWFG G is nonredundant.*

Proof. The proof is immediate. □

A key concept for proving the grammar learnability (see Sections 5.1.2 and 5.3.1.4) is the reduced grammar semantics defined below.

Definition 17. *Let G^* be a LWFG, and let E_σ be a sublanguage of G^*, $E_\sigma \subseteq L_\sigma(G^*)$. Given a LWFG 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 18. *A chain is a set of ordered unary branching rules: $\{B_k \to B_{k-1}, \ldots, B_2 \to B_1, B_1 \to \beta\}$ such that $L_\sigma(B_k) \supset L_\sigma(B_k \to B_{k-1}) \equiv L_\sigma(B_{k-1}) \supset \cdots \supset L_\sigma(B_2 \to B_1) \equiv L_\sigma(B_1) \supset L_\sigma(B_1 \to \beta)$.*[2] *A chain set for a nonterminal B_i, $0 \leq i \leq k$, is $chs(B_i) = \{B_k, \ldots, B_1, B_0\}$, where $B_k \succ B_{k-1} \cdots \succ B_1$ and $B_0 = \beta$. A grammar rule $r^+ = (A \to \beta^+\colon \Phi^+)$ is a generalized rule of a grammar rule*

[1] The grammar cannot contain unused rules since $S \succeq A, \forall A \in N_G$ (see Definition 5, (vi), pg. 51). Also, every nonterminal symbol is a left-hand side in at least one rule (see Definition 5, (v)).

$r = (A \to \beta\colon \Phi)$, 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 $r^- = (A \to \beta^-\colon \Phi^-)$ is a specialized rule of a grammar rule $r = (A \to \beta\colon \Phi)$, 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 19 (Normalized). *A Lexicalized Well-Founded Grammar G is called normalized (NLWFG) if for all grammar rules we have that $|\beta|$ is minim, i.e., $\forall A \to \beta\colon \Phi, \nexists$ a rule $A' \to \beta'\colon \Phi'$ with $(\beta' \subset \beta) \wedge (|\beta'| > 1)$.*[3]

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

Definition 20 (Conformal). *A Normalized Lexicalized Well-Founded Grammar G is conformal 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 (E_σ is a representative sublanguage, i.e., $E_\sigma \supseteq E_R$).*

Lemma 5. *Given a Normalized Lexicalized Well-Founded Grammar, G conformal w.r.t. 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^*:

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

[2]For simplicity, we use the notation $B_k \to B_{k-1}$ for the ordered unary branching rules $B_k(\sigma_k) \to B_{k-1}(\sigma_{k-1})\colon \Phi_k(\bar{\sigma})$, 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 $\forall i, j, i \neq j, L_\sigma(B_i) \not\equiv L_\sigma(B_j)$.

[3]This is a syntactic definition.

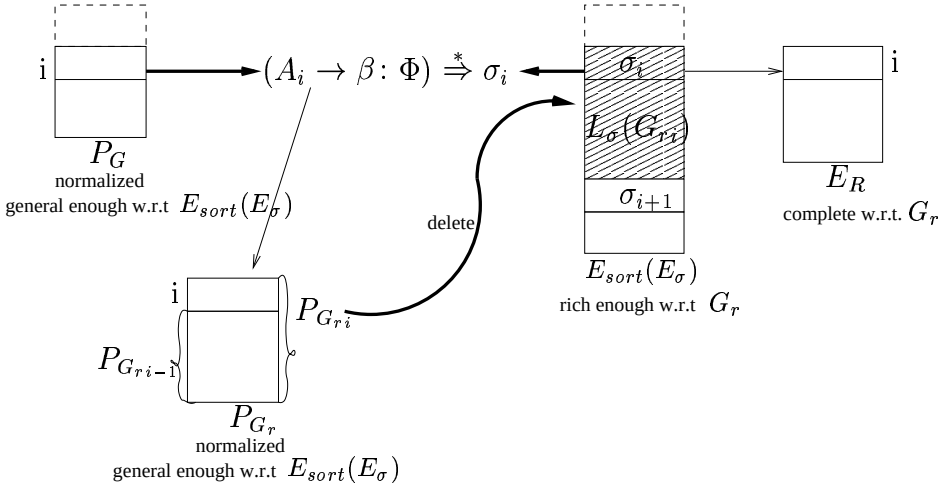


Figure 5.1: Step i in Algorithm 4. The lines in bold show that the rule associated with the minimum representative example σ_i is determined by P_G, and that σ_i belongs to $L_\sigma(G_{ri})$, which is deleted from the sublanguage E_σ.

with the following change of rules:

	G *rule*		G^* *rule*
(5.1a)	$\begin{cases} B_{j+1} \to B_j \\ B_j \to B_{j-1} \end{cases}$	becomes specialized	$\begin{matrix} B_{j+1} \to B_i \\ B_j \to B_{i-1} \end{matrix}$
(5.1b)	$\begin{cases} B_{i+1} \to B_i \\ B_i \to B_{i-1} \end{cases}$	becomes generalized	$\begin{matrix} B_{i+1} \to B_j \\ B_i \to B_{j-1} \end{matrix}$

The first two rules (5.1a) are specialized rules and the last two rules (5.1b) 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 5.3.1.4) we assume our target grammar to be a NLWFG conformal w.r.t. a representative 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 5.1, 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 $r = A_i \rightarrow \beta : \Phi$ is determined from P_G and σ_i, and then added to P_{G_r}. Since G is normalized and conformal 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 belong 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, 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 5.4). The learning algorithm is presented in Section 5.1.2, and the learnability theorem in Section 5.3.1.4.

For a Normalized Lexicalized Well-Founded Grammar, G, conformal 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

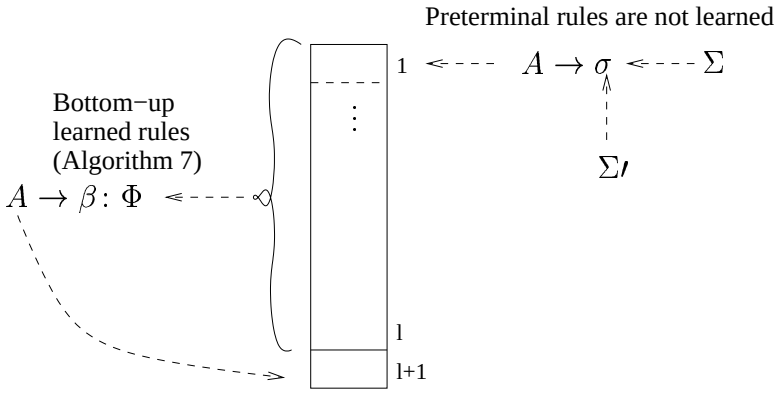


Figure 5.2: Grammar nonterminal levels. Preterminals are on level 1 and their rules are not learned.

will learn first the ordered non-recursive rules (shown in Figure 5.2), then the ordered recursive rules and last the non-ordered rules (see Algorithm 7). In the absence of this ordering, the learning machinery might need theory revision steps (see Algorithm 9).

Table 5.1 presents a summary of the main properties of Normalized Lexicalized Well-Founded Grammars and their consequences for learning. The first property, well-foundedness 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 5.2, 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 ensure 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 Pro-

Properties of LWFGs	**Consequences for Learning**
N_G is well founded	Bottom-up induction of the grammar
Every nonterminal is a left-hand side in at least one ordered non-recursive rule	Termination condition for parsing[a] $\Rightarrow$ ILP decidability
ϵ cannot be derived from any nonterminal	
$\forall\sigma, A \stackrel{*}{\Rightarrow} \sigma$ have the same category of their semantic molecules	Predicate invention for induction
Ground syntagma derivation $A \stackrel{*G}{\Rightarrow} \sigma$	Grammar-provability $K \wedge G \vdash_{rp} A(\sigma)$
Representative examples	Small semantic treebank
Representative sublanguage	Defines the reduced semantics
G conformal to the representative sublanguage	Learnability from positive examples

[a]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.

Table 5.1: Properties of NLWFGs and their implications for learning

gramming 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 5.1.2, (Muresan, Muresan, and Potolea, 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 two properties allow us to learn only from positive data, which is essential, given that negative evidence is rarely available in language learning.

5.1.2 Relational Learning from Ordered Representative Examples

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 and De Raedt, 1994; De Raedt, 1996; Lloyd, 2003). ILP methods have been used in a variety of applications for natural language processing (Zelle and Mooney, 1993; Adriaans and Haas, 1999; Cussens and Džeroski, 2000; Tang and Mooney, 2001) and relational data mining, including applications for bioinformatics (Džeroski and Lavrač, 2001).

Our learning algorithm for grammar induction is based on our previous work (Muresan, Muresan, and Potolea, 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-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 molecules, and 5) a reversible robust parser as innate inference engine. Initially, (1) and (2) can be the empty set, $\emptyset$.

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.1.1 and Appendix A.4.1 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 ensure unambiguity.

Algorithm 7 describes the constraint-based grammar induction based only on positive examples.

Algorithm 7: Constraint_Grammar_Induction(E_R, E_σ, K)

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

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 5.2, and Figure 5.3).

In step 1 of the Generate_Rule procedure, the robust parser generates the

Procedure `Generate_Rule`$(\sigma, G', E_\sigma, K)$

$\sigma = (w, \binom{h}{b})$

1 $w \leftarrow \min(w_1 \ldots w_n)$ s.t $b = (b_1, \ldots, b_n)\nu$ /*$w = w_1 \ldots w_n$, $(w_j, h_j \bowtie b_j) \in L_\sigma(w, b)$, $1 \leq j \leq n$; n is the minimun number of chunks given by robust parser */

2 $chs(j) = \{B_j | \sigma_j \equiv (w_j, h_j \bowtie b_j), B_j \overset{*G'}{\Rightarrow} \sigma_j\}, 1 \leq j \leq n$ /*by robust parser */

$chain(j) = \{B_j^{K_j}, B_j^{K_j - 1}, \ldots, B_j^2, B_j^1, B_j^0\}$ /*ordered chain rules, $B_j^0 = \beta_j$ */

3 r $\leftarrow (A \rightarrow B_1^0, \ldots, B_n^0 : \Phi_r)$

for $j \leftarrow 1$ ***to*** n **do**

  $i \rightarrow 1$

  while $r \overset{B_j^i \rightarrow B_j^{i-1}}{\dashv} r_g \wedge \mathbb{S}(r) \subset \mathbb{S}(r_g)$ **do**

    $i \rightarrow i + 1$

    $r \leftarrow r_g$

return r

minimum number of chunks that cover σ (starting from the string w of σ)[4]. 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 5.4). In step 3, the most specific rule r, i.e, with minimum reduced semantics 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^0 from its right-hand side are generalized (see assumption A2). Then, the rule r is generalized by unary branching rules as long as the reduced semantics increases. We denote by $r \overset{B_j^i \rightarrow B_j^{i-1}}{\dashv} r_g$ the generalization of r by unary branching rule $B_j^i \rightarrow B_j^{i-1}$ in chain(j) (see section 5.3.1). Therefore, in our Inverse Entailment learning method, the reduced grammar semantics is used as the performance criterion for selecting the final hypothesis r. The final rule r is the least generalized rule which cannot produce a semantic

[4]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.

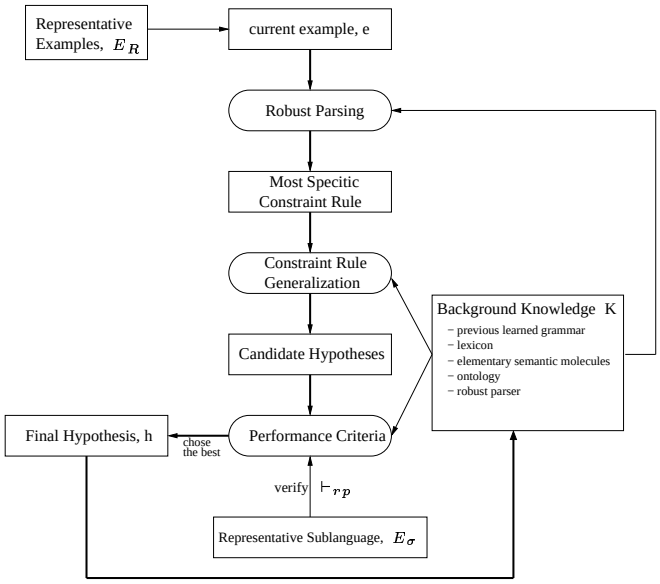


Figure 5.3: An iteration of Algorithm 7

increase. It is guaranteed that the rule r is normalized and general enough w.r.t. E_σ, in accord with assumption A4. An example of an iteration step is given in Figure 5.3, and in Appendix A.2.

The algorithm is linear on the length of the learned hypothesis and has the complexity $O(|E_R| * |\beta| * |chs(j)| * |E_\sigma| * |\sigma|^3)$. We assume a constant bound on the length of the grammar rules. Examples of learned grammars and constraints are given in Appendix A.1.1 and Appendix A.4.

Theorem 2 (The NLWFG Induction Theorem). *Given a Normalized Lexicalized Well-Founded grammar, G, conformal to a representative 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 7 generates a grammar G' s.t. $G' = G$. We write $(E_R, E_\sigma) \xrightarrow{7} G$.*

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 $A_i \rightarrow B_1, \ldots, B_n : \Phi \overset{*}{\Rightarrow} \sigma_i$ and the rule $A_i \rightarrow B_1, \ldots, B_n : \Phi$ is normalized and general

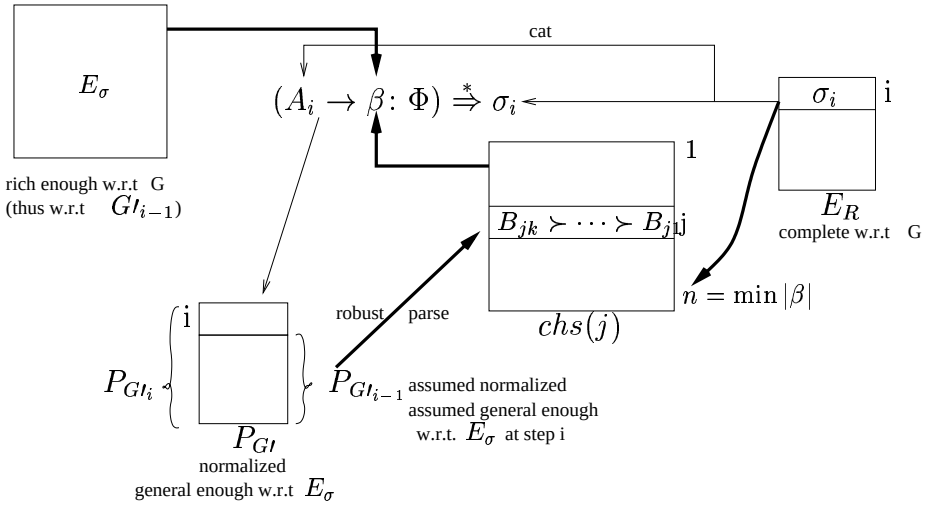


Figure 5.4: Step i in Algorithm 7. The lines in bold show that the rule rhs, β, is learned using the previous $i-1$ learned rules, $P_{G'_{i-1}}$, the current example, σ_i, and the sublanguage E_σ.

enough w.r.t. E_σ (n is minimum (Figure 5.1)). Since G is conformal to E_σ, it follows that Algorithm 7 (which guarantees that the learned rules are normalized and general enough w.r.t. E_σ), computes for σ_i, exactly the same rule $A_i \rightarrow B_1, \ldots, B_n\colon \Phi \stackrel{*}{\Rightarrow} \sigma_i$ at step i, and thus $P_{G'_i} = P_{G_i}$ (Figure 5.4). By complete induction, it follows that $G' = G$. □

5.1.3 Iterative Learning from Unordered Representative Examples

Algorithm 7 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 9 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. The reduced semantics of

all the rules/chains relative to E_σ is non-decreasing.

Algorithm 9: Iterative_Grammar_Induction(E_R^u, E_σ, K)

```
P_G' ← ∅
repeat
    O_G' ← P_G'
    for i ← 1 to |E_R^u| do
        σ_i ← E_R^u(i)          /*unordered set of representative examples */
        r = (A → β: Φ) s.t σ_i ∈ L_σ(r)
        P_G' ← P_G' − {r}
        r ← Generate_Rule(σ_i, G', E_σ, K)  /*regenerates the ith rule based on
                                              all the other rules */
        P_G' ← P_G' ∪ {r}
until O_G' = P_G'
return P_G'
```

Theorem 3. *Given a Normalized Lexicalized Well-Founded Grammar G conformal 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 9 learns the same grammar as Algorithm 7 would do, if provided with the representative example set E_R in the right order. We write $(E_R^u, E_\sigma) \xrightarrow{9} G$ iff $(E_R, E_\sigma) \xrightarrow{7} G$.*

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 Rule Generalization Property (Property 4) of semantic composition (Section 4.1.4.2, pg. 84), and Lemma 5 (Section 5.1.1), 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 9, i.e., $(E_R^u, E_\sigma) \xrightarrow{9} G$. The grammar G is conformal to E_σ [5], which implies that $(G, E_\sigma) \xrightarrow{4} (E_R, G)$ (Theorem 1), and thus $(E_R, E_\sigma) \xrightarrow{7} G$ (Theorem 2). Proving the reciprocal is immediate, since it is sufficient to take E_R^u in the right order in Algorithm 9. □

In Appendix A.1.2, we show a run of the iterative algorithm for learning finite auxiliary verbs from unordered examples. We show the iterations steps and the reduced semantics at each step (i.e., the number of covered positive examples from the representative sublanguage used for generalization).

[5] The Generate_Rule procedure guarantees that the learned rules are normalized and general enough w.r.t. E_σ.

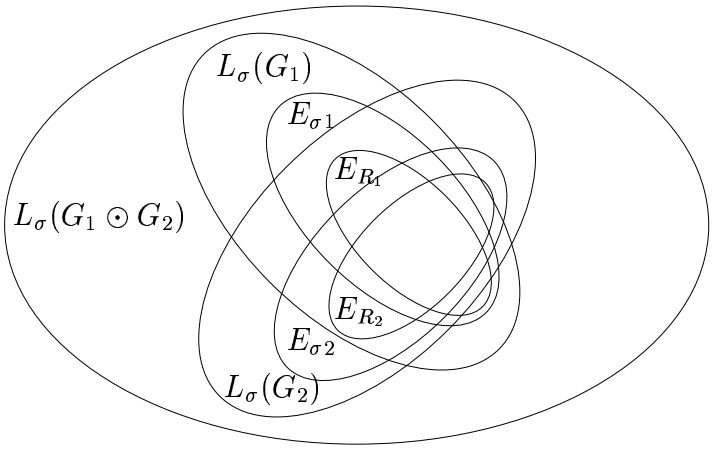


Figure 5.5: Merging two NLWFGs G_1 and G_2

5.2 Grammar Merging

One of the major concerns regarding grammar development and engineering is how to formally define grammar modularity, such that different fragments of grammars can be combined together in a sound way (Wintner, 2002). In this section we show how LWFGs 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.

In the previous sections, it could be noticed that for a Normalized Well-Founded Grammar G conformal to a sublanguage E_σ, Algorithm 4 and Algorithms 7/9 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 $G \overset{E_\sigma}{\longleftrightarrow} E_R$. The direction $\rightarrow$ is given by Algorithm 4 and the direction $\leftarrow$ by Algorithm 7/9, respectively.

Definition 21 (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 $N_{G_1} \subseteq N_G$, $N_{G_2} \subseteq N_G$*

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

where R_{G_1} and R_{G_2} are consistent with each other.[6] *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.

(ii) The union of the sets of representative examples, $E_{R_1} \cup E_{R_2}$,[7] *and the union of the sublanguages, $E_{\sigma 1} \cup E_{\sigma 2}$ are performed.*

(iii) Algorithm 7 or Algorithm 9 is 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 ensures that: $L_\sigma(G_1) \cup L_\sigma(G_2) \subseteq L_\sigma(G_1 \odot G_2)$ (see Figure 5.5).*

Proof. The proof is immediate. □

[6] Between the common nonterminals there is no contradictory partial ordering relation.

[7] 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 9 could iterate only the rules that corresponds to nonterminals that belong to the cut nonterminal set, increasing the efficiency.

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 5.6: 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 $h.a$.

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 5.6. 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 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 5.7, 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

	G_1	G_2	$G = G_1 \odot G_2$
P_G	 $N_2 \rightarrow Det\ Noun : \Phi_4$ $N_2 \rightarrow Pn : \Phi_5$ $C_1 \rightarrow N_2\ Tv\ N_2 : \Phi_6$	$N_1 \rightarrow Noun : \Phi_1$ $N_1 \rightarrow Adj\ N_1 : \Phi_2$ $N_2 \rightarrow Det\ N_1 : \Phi_3$	$N_1 \rightarrow Noun : \Phi_1$ $N_1 \rightarrow Adj\ N_1 : \Phi_2$ $N_2 \rightarrow Det\ N_1 : \Phi_3$ $N_2 \rightarrow Pn : \Phi_5$ $C_1 \rightarrow N_2\ Tv\ N_2 : \Phi_6$
Σ	$Noun \rightarrow [child]$ $Noun \rightarrow [day]$ $Adj \rightarrow [sunny]$ $Adj \rightarrow [playful]$ $Tv \rightarrow [like]$ $Det \rightarrow [the]$ $Det \rightarrow [a]$ $Pn \rightarrow [john]$	$Noun \rightarrow [child]$ $Noun \rightarrow [day]$ $Adj \rightarrow [sunny]$ $Adj \rightarrow [playful]$ $Tv \rightarrow [like]$ $Det \rightarrow [the]$ $Det \rightarrow [a]$ $Pn \rightarrow [john]$	$Noun \rightarrow [child]$ $Noun \rightarrow [day]$ $Adj \rightarrow [sunny]$ $Adj \rightarrow [playful]$ $Tv \rightarrow [like]$ $Det \rightarrow [the]$ $Det \rightarrow [a]$ $Pn \rightarrow [john]$
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)

$E_{\sigma 1}$	$E_{\sigma 2}$	$E_{\sigma 1} \cup E_{\sigma 2}$
	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)

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

(c)

Figure 5.7: Merging two grammars G_1 and G_2

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 Chapter 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)).

5.3 Grammar Approximation by Representative Sublanguage (GARS)

In this section we present our new relational learning model for LWFG learning, *Grammar Approximation by Representative Sublanguage (GARS).* For this, we first present the theoretical foundation of the hypothesis search space for LWFG learning.

5.3.1 Foundation of the Search Space for Grammar Induction

In Theorem 2 and Theorem 3 we have proved the soundness of LWFG learning, done either from ordered representative examples (Algorithm 7), or from unordered representative examples (Algorithm 9). In this section we prove that the hypothesis search space for any LWFG learning algorithm is a complete grammar lattice, and give a learnability theorem. The results of this section are an extension of our paper

(Muresan, 2005).

5.3.1.1 Representative Examples Parsing Preserving Grammars

In order to define the search space of grammar induction as grammar lattice, we define *the rule derivation 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 derived/generalized rules ground derive the same syntagma, $\sigma_A \in E_R$.

Definition 22. *The rule derivation step:*

$$\frac{A(\sigma_A)\rightarrow\alpha B(\sigma_B^*)\gamma\colon \Phi_A \qquad B(\sigma_B)\rightarrow\beta\colon \Phi_B}{A(\sigma_A)\rightarrow\alpha\beta\gamma\colon \Phi'_A} \tag{5.2}$$

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

The rule generalization step :

$$\frac{A(\sigma_A)\rightarrow\alpha\beta\gamma\colon \Phi'_A \qquad B(\sigma_B)\rightarrow\beta\colon \Phi_B}{A(\sigma_A)\rightarrow\alpha B(\sigma_B^*)\gamma\colon \Phi_A} \tag{5.3}$$

is E_R parsing preserving, if $r'_A \overset{*G'}{\Rightarrow} \sigma_A \wedge r_A \overset{*G}{\Rightarrow} \sigma_A \wedge \sigma_A \in E_R$. We write $r'_A \overset{r_B}{\dashv} r_A$.

Since σ_A is a representative example, it has the minimum ground derivation length ($gdl(\sigma_A) = mgdl(r_A)$) and thus, we have that r_B is an ordered non-recursive rule. The goal of the rule derivation 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. An example of rule derivation steps is given in Figure 5.8. The derivation step $r_2 \overset{r_1}{\vdash} r'_2$ is E_R parsing preserving, because the rule r'_2 ground derives the syntagma *major damage*.

G	G'	G''	E_R
$r_1\ N \rightarrow Noun$	$r_1\ N \rightarrow Noun$	$r_1\ N \rightarrow Noun$	damage
$r_2\ N \rightarrow Adj\ \ N$	$r'_2\ N \rightarrow Adj\ Noun$	$r''_2\ N \rightarrow Adj\ Adj\ N$	major damage

Figure 5.8: Example and Counterexample of E_R parsing preserving. Derivation step $r_2 \overset{r_1}{\vdash} r'_2$ is E_R parsing preserving, while the derivation step $r_2 \overset{r_2}{\vdash} r''_2$ is not

The derivation step $r_2 \overset{r_2}{\vdash} r''_2$ is not E_R parsing preserving since the syntagma *major damage* cannot be ground-derived from the rule r''_2.

We assume that the compositional constraint of the resulting rule (Φ'_A for derivation, and Φ_A for generalization) is computed with the Algorithm 6 (Generate_Compositional_Constraints). Thus, for both specialization and generalization substitutions, ν and μ, respectively, the same relation hold for both rule derivation and generalization steps (see (4.18) in Chapter 4, Section 4.1.4.2):

$$\begin{aligned} \nu'_A &= \nu_{A(B)} \\ \mu'_A &\subseteq \mu_{A(B)} \end{aligned} \tag{5.4}$$

In other words, the specialized rule r'_A can have the contextual constant generalization substitution μ incomplete. This is explained by Principle 5 (Feature Agreement Principle, Chapter 4, pg. 73) where the agreement is done only at the level of the semantic head.[8]

From the Rule Generalization Property (Property 4 in Section 4.1.4.2, pg. 84), we have that:

$$\mathrm{L}_\sigma(r_A) \supseteq L_\sigma(r'_A). \tag{5.5}$$

[8]We emphasize that all properties given in this section still hold even if the μ substitution is absent (i.e., we do not have contextual constant generalization).

Definition 23. *A grammar G' is one-step derived from a grammar G, $G \overset{r_1}{\vdash} G'$, if $\exists r, r_1 \in P_G \wedge \exists r', r_1 \in P_{G'}$, 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 derived from a grammar G, $G \overset{*}{\vdash} G'$, if it is obtained from G in n-derivation steps: $G \overset{r_1}{\vdash} \cdots \overset{r_n}{\vdash} G'$, where n is finite. We extend the notation so that we have $G \overset{*}{\vdash} G$.*

In Figure 5.8, the grammar G' is one-step derived from the grammar G, i.e., $G \overset{r_1}{\vdash} G'$, since G' preserve the parsing of the representative examples E_R, as we discussed above. The grammar G'' is not derived from the grammar G since it does not preserve the representative example set anymore, as we have shown.

Definition 24. *A grammar G is one-step generalized from a grammar G', $G' \overset{r_1}{\dashv} G$, if $\exists r, r_1 \in P_G \wedge \exists r', r_1 \in P_{G'}$, s.t. $r' \overset{r_1}{\dashv} r$, and $\forall q \neq r, q \in P_G$ iff $q \in P_{G'}$. A grammar G is generalized from grammar G', $G' \overset{*}{\dashv} G$, if it is obtained from G' in n-generalization steps: $G' \overset{r_1}{\dashv} \cdots \overset{r_n}{\dashv} G$, where n is finite. We extend the notation so that we have $G \overset{*}{\dashv} G$.*

Definition 25 (Normalized). *A LWFG G is called normalized w.r.t. a sublanguage $E_\sigma \subseteq L_\sigma(G)$, if all grammar rules cannot be further generalized by the rule generalization step (5.3), such that $\mathbb{S}(r'_A) \subset \mathbb{S}(r_A)$.*[9]

Definition 26. *Let $\top$ be a LWFG, normalized and unambiguous w.r.t. a sublanguage $E_\sigma \subseteq L_\sigma(\top)$, and let $E_R \subseteq E_\sigma$ be its set of representative examples. Let $\mathcal{L} = \{G | \top \overset{*}{\vdash} G\}$ be the set of grammars derivable from $\top$. We call $\top$ the top element of $\mathcal{L}$, and $\bot$ the bottom element of $\mathcal{L}$, if $\forall G \in \mathcal{L}, \top \overset{*}{\vdash} G \wedge G \overset{*}{\vdash} \bot$. The bottom element, $\bot$, is the grammar derived from $\top$, such that the right-hand side of all grammar rules contains only preterminals. We have $\mathbb{S}(\top) = E_\sigma$ and $\mathbb{S}(\bot) \supseteq E_R$ (see Figure 5.15(a)).*

Lemma 6. *For $G, G' \in \mathcal{L}$, $G \overset{*}{\vdash} G'$ iff $G' \overset{*}{\dashv} G$ and $L_\sigma(G) \supseteq L_\sigma(G')$.*

Proof. Immediate from Property 4, see (5.5). □

Lemma 7. *$\forall G \in \mathcal{L}$ and $\forall \sigma \in E_R$, $\exists r \in P_G$, s.t. $r \overset{*}{\Rightarrow} \sigma$.*

[9] This is a semantic definition.

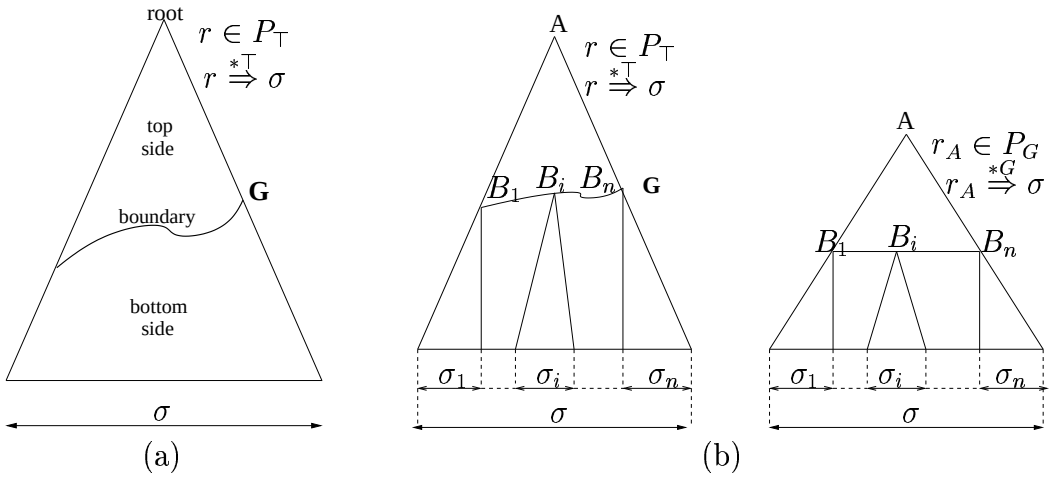


Figure 5.9: (a) Grammar boundary ; (b) Subtree correspondence

Proof. The property holds for $\top$ and $\top \overset{*}{\vdash} G$ is E_R parsing preserving. □

Definition 27. *If $G, G' \in \mathcal{L}$, we say that G subsumes G', i.e., $G \succcurlyeq G'$, iff $G \overset{*}{\vdash} G'$.*

Theorem 5. *For $G, G' \in \mathcal{L}$, if $G \succcurlyeq G'$ then $\mathbb{S}(G) \supseteq \mathbb{S}(G')$.*

Proof. From Lemma 6, Definition 17 (Section 5.1.1), and Definition 27. □

Definition 28. *We call boundary of a grammar $G \in \mathcal{L}$ relative to the parse tree $r \overset{*\top}{\Rightarrow} \sigma$,[10] the right-hand side of the corresponding rule $r_A \in P_G, r_A \overset{*G}{\Rightarrow} \sigma$: $bd(G) = \{B | r_A \in P_G, B \in rhs(r_A)\}$[11] (see Figure 5.9(a)). We denote by $f_N(r \overset{*\top}{\Rightarrow} \sigma)$ the set of nonterminals which belong to the parse tree of $r \overset{*\top}{\Rightarrow} \sigma$, where f_N: $P_\top \times E_R \rightarrow N_\top, r \in P_\top, \sigma \in E_R$. We call top-side, $ts(G)$, and respectively bottom-side, $bs(G)$, of grammar G relative to the parse tree $r \overset{*\top}{\Rightarrow} \sigma$, the sets of the nonterminals delimited by G boundary, $bd(G)$ (see Figure 5.9(a)):*

$$ts(G) = \{B \in f_N(r \overset{*\top}{\Rightarrow} \sigma) | \exists B_i \in bd(G) \wedge B \succeq B_i\} \cup \{root\}$$
$$bs(G) = \{B \in f_N(r \overset{*\top}{\Rightarrow} \sigma) | \exists B_i \in bd(G) \wedge B \preceq B_i\}^{12}$$

---

[10] All grammars G, $\top \overset{*}{\vdash} G$, are E_R parsing preserving and all boundaries of G are in the parse trees of the ground derivations of $\top$ grammar rules.

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

[12] $\succeq$ is the partial ordering relation among the LWFG nonterminals.

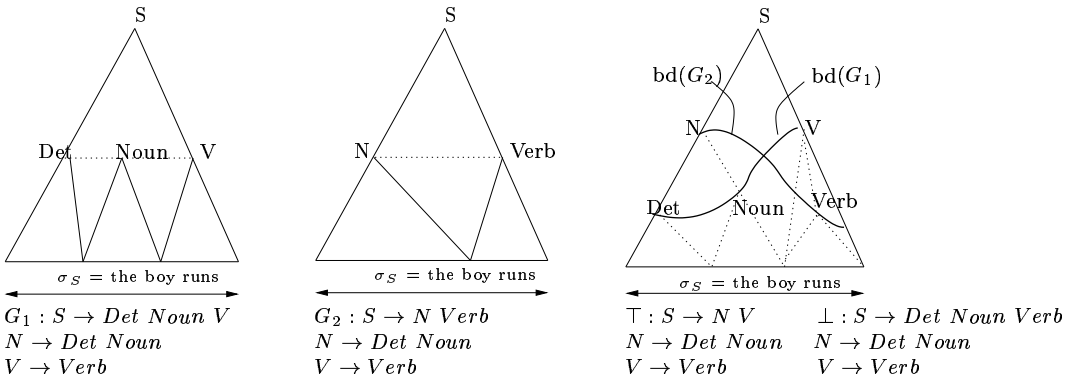


Figure 5.10: Grammars that preserve the parsing of the syntagma *the boy runs*

We have that $ts(G) \cap bs(G) = bd(G)$, $ts(G) \cup bs(G) = f_N(r \overset{*\top}{\Rightarrow} \sigma)$ and for the top element of $\mathcal{L}$: $ts(\top) = bd(\top) \cup \{root\}$.

In Figure 5.10 we present a concrete example. The grammars G_1, G_2, $\top$, $\bot$ preserve the parsing of syntagma *the boy runs*, and we have that $\top \overset{r_1}{\vdash} G_1$, $\top \overset{r_2}{\vdash} G_2$, $G_2 \overset{r_1}{\vdash} \bot$, $G_1 \overset{r_2}{\vdash} \bot$ and $\top \overset{*}{\vdash} \bot$. We can also see the boundaries of the grammar G_1 and G_2, relative to the parse tree of the $\top$ grammar (the right-most figure).

Lemma 8. *$\forall G \in \mathcal{L}, \forall r_A \in P_G, r_A \overset{*G}{\Rightarrow} \sigma$, and $\forall B_i \in rhs(r_A)$, the parse tree $B_i \overset{*G}{\Rightarrow} \sigma_i$ has a corresponding subtree in the parse tree $r \overset{*\top}{\Rightarrow} \sigma$, rooted at the same nonterminal $B_i \in bd(G)$, such that $B_i \overset{*\top}{\Rightarrow} \sigma_i$.*

Proof. The property holds due to the unambiguity of the $\top$ grammar and the E_R *parsing preserving property* of the rule derivation step. Moreover, the rule derivation step preserves grammar unambiguity. If $r_A \in P_G$ is $A \to B_1, \ldots, B_n$, we have that $\sigma = \sigma_1 \cdots \sigma_n$ in both parse trees $r_A \overset{*G}{\Rightarrow} \sigma$ and $r \overset{*\top}{\Rightarrow} \sigma$ (see Figure 5.9(b)). □

Lemma 9. *$\forall G, G' \in \mathcal{L}$, $\forall r_A \in P_G$ and $\forall r'_A \in P_{G'}$, with $r_A \overset{*G}{\Rightarrow} \sigma$ and $r'_A \overset{*G'}{\Rightarrow} \sigma$, $\sigma \in E_R$, if $B \in rhs(r_A), B' \in rhs(r'_A)$ and $B \overset{*G}{\Rightarrow} \sigma_B$,$B' \overset{*G'}{\Rightarrow} \sigma'_B$, then $\sigma_B \subseteq \sigma'_B \vee \sigma_B \supseteq \sigma'_B \vee \sigma_B \cap \sigma'_B = \emptyset$.*

Proof. $\top$ is a normalized and unambiguous LWFG, and $r \in P_\top$ has a unique parse tree $r \overset{*\top}{\Rightarrow} \sigma$. Since both G and G' are grammars derived from $\top$, the parse trees $B \overset{*G}{\Rightarrow} \sigma_B$ and $B' \overset{*G'}{\Rightarrow} \sigma'_B$ have corresponding subtrees in $r \overset{*\top}{\Rightarrow} \sigma$, which have the same root due to grammar unambiguity: $B \overset{*\top}{\Rightarrow} \sigma_B$ and $B' \overset{*\top}{\Rightarrow} \sigma'_B$, respectively (Lemma

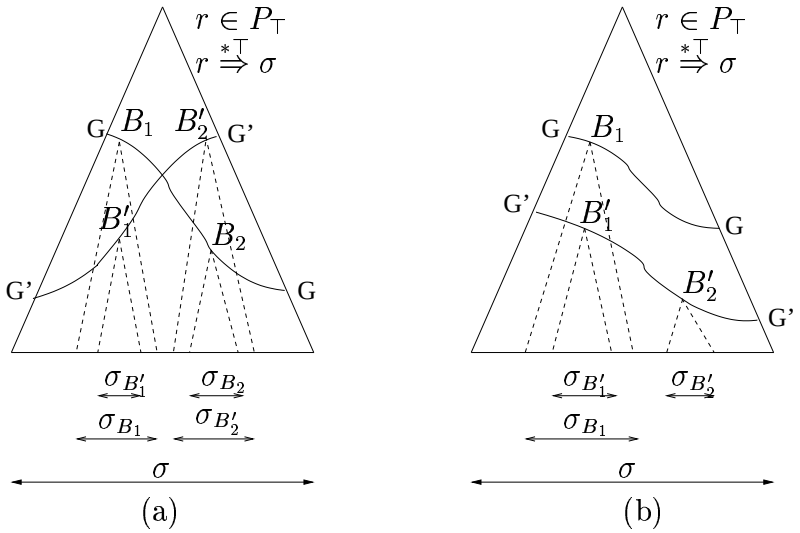


Figure 5.11: Subsyntagma relations

8). Since no two subtrees of a tree overlap in an unambiguous grammar, the lemma property holds (Figure 5.11(a)). □

Lemma 10. *Given the same conditions as in Lemma 9, if* $G \overset{*}{\vdash} G'$*, then* $\sigma_B \supseteq \sigma'_B \vee \sigma_B \cap \sigma'_B = \emptyset$.

Proof. The proof is similar to the one for Lemma 9 (see Figure 5.11(b)). □

5.3.1.2 Semantic-based Complete Grammar Lattice

We consider the system $\mathfrak{L} = \langle \mathcal{L}, \succcurlyeq \rangle$ formed by the set $\mathcal{L}$ of the grammars derivable from $\top$, together with the binary subsumption relation $\succcurlyeq$ that establishes a partial order in $\mathcal{L}$. In order for this system to form a lattice, we must define two operators: the *least upper bound* (*lub*), $\curlyvee$ and the *greatest lower bound* (*glb*), $\curlywedge$, such that for any two elements $G_1, G_2 \in \mathcal{L}$, the elements $G_1 \curlyvee G_2, G_1 \curlywedge G_2 \in \mathcal{L}$ exist (Tarski, 1955). The *lub* element of G_1, G_2 is the minimum element that has the boundary above the boundaries of G_1 and G_2. The *glb* element of G_1, G_2 is the maximum element that has the boundary below the boundaries of G_1 and G_2. Thus, *lub* and

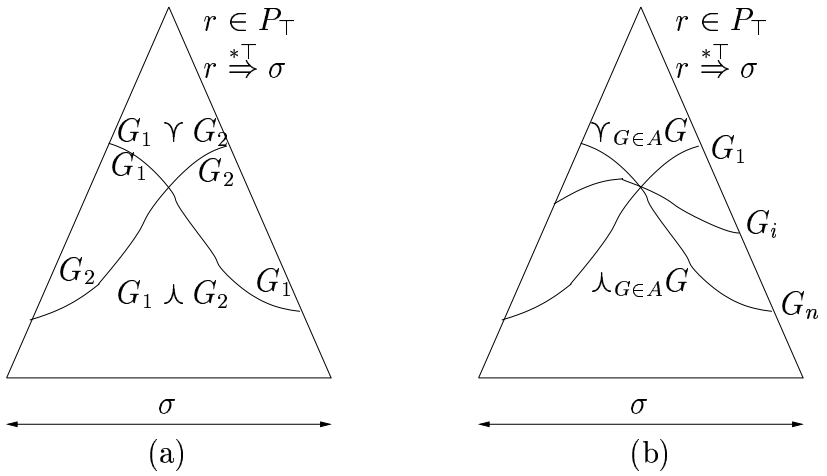


Figure 5.12: The *lub* and *glb* operators

glb are defined such that for all grammar rules we have:

$$\begin{aligned} ts(G_1 \curlyvee G_2) &= ts(G_1) \cap ts(G_2) \\ bs(G_1 \curlywedge G_2) &= bs(G_1) \cap bs(G_2) \end{aligned} \tag{5.6}$$

as can be seen in Figure 5.12(a). For the example in Figure 5.10, we see that $\top = G_1 \curlyvee G_2$, $\bot = G_1 \curlywedge G_2$. In order to have a complete lattice, the property must hold $\forall A \subseteq \mathcal{L}$:

$$\begin{aligned} ts(\curlyvee_{G \in A} G) &= \bigcap_{G \in A} ts(G) \\ bs(\curlywedge_{G \in A} G) &= \bigcap_{G \in A} bs(G) \end{aligned} \tag{5.7}$$

as can be seen in Figure 5.12(b).

These two operators are defined by Algorithms 10 and 11, which generate the rules corresponding to the grammars $G_1 \curlyvee G_2$, and $G_1 \curlywedge G_2$ based on the corresponding rules in G_1 and G_2 and the operators $\curlyvee$ and $\curlywedge$.

[13] We need to include the statement if $x_2 \succ x_1$ then $x_1 \longleftrightarrow x_2$ iff chains of unary branching rules are considered.

Algorithm 10: Least_Upper_Bound(G_1, G_2)

```
for i ← 1 to |E_R| do
    P_{G1 ϒ G2}(i) ← P_{G1}(i) ϒ P_{G2}(i)   /*ϒ(G1, G2, i)                    */
    return P_{G1 ϒ G2}
```

Algorithm 11: Greatest_Lower_Bound(G_1, G_2)

```
for i ← 1 to |E_R| do
    P_{G1 ⅄ G2}(i) ← P_{G1}(i) ⅄ P_{G2}(i)   /*⅄(G1, G2, i)                    */
return P_{G1 ⅄ G2}
```

Procedure `Op`(G_1, G_2, i)

```
/*   P_{G1}(i) ϒ P_{G2}(i) or P_{G1}(i) ⅄ P_{G2}(i)                          */
l ← lhs(P_{G1}(i))   /*   = lhs(P_{G2}(i))                                  */
σ ← E_R(i)
r_{x1} ← rhs(P_{G1}(i)))    r_{x2} ← rhs(P_{G2}(i)))
r_ϒ ← r_⅄ ← ∅
x1 ← next(r_{x1})    x2 ← next(r_{x2})
while x1 ≠ ∅ ∨ x2 ≠ ∅ do
    σ_{x1} ← σ1  s.t.  σ1 ⊆ σ  and  x1 ⇒*G1 σ1
    σ_{x2} ← σ2  s.t.  σ2 ⊆ σ  and  x2 ⇒*G2 σ2
    if σ_{x1} ⊆ σ_{x2} ∨ σ_{x1} ⊇ σ_{x2} then
1       if σ_{x1} ⊃ σ_{x2} then
            r_ϒ ← r_ϒ@ x1(σ_{x1})    r_⅄ ← r_⅄@ x2(σ_{x2})
                /*@ is the concatenation operator                           */
2       if σ_{x1} = σ_{x2} then
            /*x1 = x2 [13]                                                  */
            r_ϒ ← r_ϒ@ x1(σ_{x1})    r_⅄ ← r_⅄@ x2(σ_{x2})
3       if σ_{x1} ⊂ σ_{x2} then
            r_ϒ ← r_ϒ@ x2(σ_{x2})    r_⅄ ← r_⅄@ x1(σ_{x1})    r_{x1} ⟷ r_{x2}
        x1 ← next(r_{x1})
        x2 ← next(r_{x2})
4   else
        r_⅄ ← r_⅄@ x2(σ_{x2})    x2 ← next(r_{x2})
Φ_ϒ ← Generate_Constraints(l → r_ϒ)
Φ_⅄ ← Generate_Constraints(l → r_⅄)
if Op = ϒ then return l → r_ϒ : Φ_ϒ else return l → r_⅄ : Φ_⅄
function next(r)
    x ← first(r)
    r ← rest(r)
    return x
```

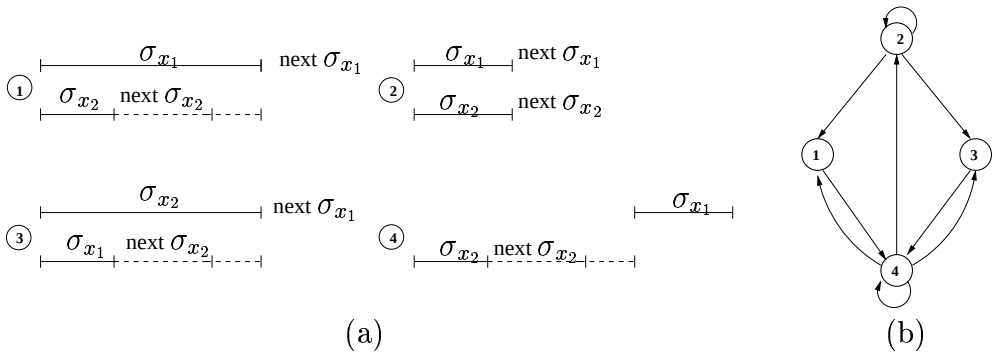


Figure 5.13: (a) Cases of syntagma relations ; (b) Transition diagram

For this, the Procedure $\mathrm{Op}(G_1, G_2, i)$ is built based on Lemma 9. The input consists of the grammar rules $P_{G_1}(i)$ and $P_{G_2}(i)$, which ground derive the same syntagma $\sigma = E_R(i)$. The index i shows the bijective mapping between the grammar rules, P_G and the representative examples, E_R. The output consists of the corresponding rules $P_{G_1 \curlyvee G_2}(i)$, and $P_{G_1 \curlywedge G_2}(i)$ which ground derive the same representative example $\sigma = E_R(i)$ (see also Figure 5.14).

The right-hand sides r_{x_1}, r_{x_2} of the input grammar rules are traversed from left to right and the corresponding right-hand sides $r_\curlyvee$, $r_\curlywedge$ of the output grammar rules are computed. For each right nonterminal x_1, x_2 of the input rules, the syntagmas $\sigma_{x_1}, \sigma_{x_2}$, which derive from them, are computed. The nonterminal whose ground-derived syntagma includes the other's syntagma, is appended to $r_\curlyvee$, while the other nonterminal is appended to $r_\curlywedge$ (see case 1 and 3 in Procedure `Op`). For the equality case (case 2), the nonterminal is appended to both rules $r_\curlyvee$ and $r_\curlywedge$. Based on Lemma 9, we have four cases in Procedure `Op`, illustrated in Figure 5.13(a), where the syntagmas in two consecutive steps are shown. The 4th case necessarily follows after case 1 and 3, where the nonterminal x_2 is appended to $r_\curlywedge$. We noticed that in case 3, r_{x_1} and r_{x_2} are swapped. The transition diagram among the 4 cases

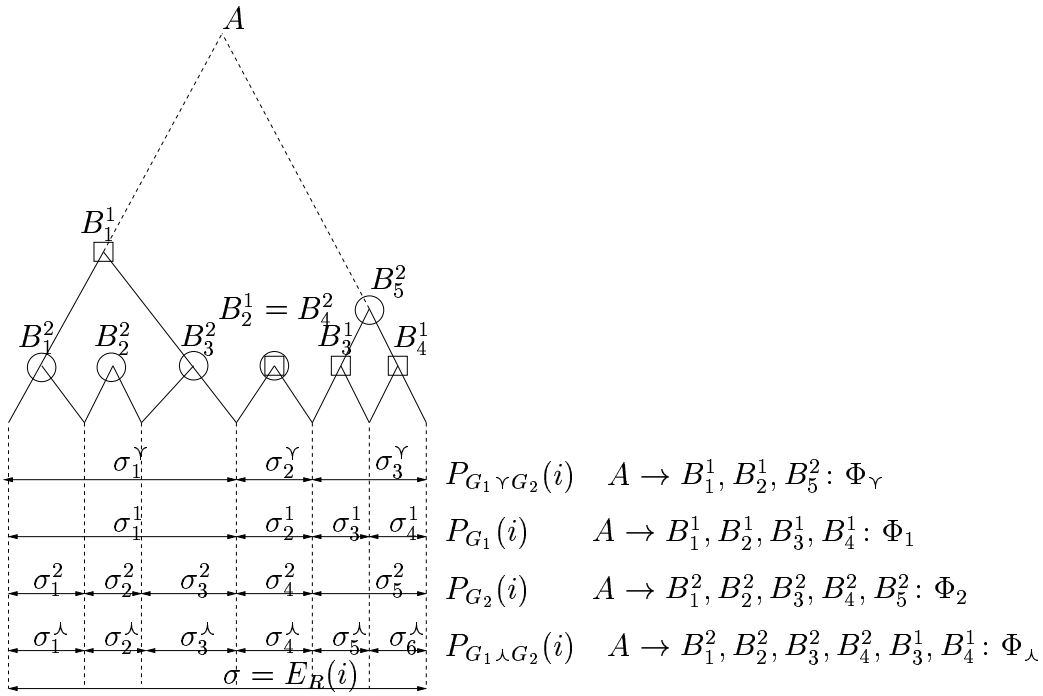


Figure 5.14: Example of computing *lub* and *glb*

is shown in Figure 5.13(b). A full cycle of Procedure `Op` is exemplified in Figure 5.14, where the ground-derived subsyntagmas are also shown.

At the end of the Procedure `Op` both $\Phi_\curlyvee$ and $\Phi_\curlywedge$ are computed, based on the corresponding rules previously computed.[14] This is in accordance with the principle that the rule derivation/generalization steps are the inverse of each other, since *lub* is a generalization, while *glb* is a derivation.

Lemma 11. *The system $\mathfrak{L} = \langle \mathcal{L}, \succcurlyeq \rangle$ together with the lub and glb operators computed by Algorithms 10 and 11, guarantees that for any two grammars $G_1, G_2 \in \mathcal{L}$ the following property holds: $G_1 \curlyvee G_2 \succcurlyeq G_1, G_2 \succcurlyeq G_1 \curlywedge G_2$ ($G_1 \curlyvee G_2 \overset{*}{\vdash} G_1, G_2 \overset{*}{\vdash} G_1 \curlywedge G_2$).*

Proof. From Procedure `Op` (Figure 5.13 and Figure 5.14), it results that the boundaries $bd(G_1 \curlyvee G_2)$ and $bd(G_1 \curlywedge G_2)$ are computed with respect to (5.6) such that the theorem property is guaranteed for each grammar rule (see Figure 5.12(a)). □

[14] The constraints can be computed based on the syntagmas which augment the grammar non-terminals (Muresan, Muresan, and Klavans, 2005, pg. 10).

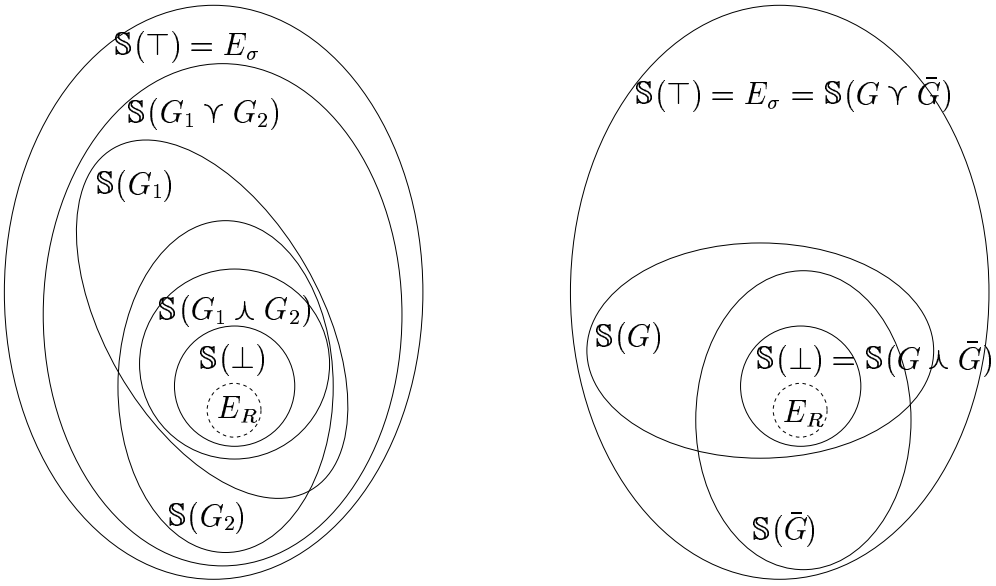


Figure 5.15: Grammar semantics reduced to the sublanguage E_σ: a) Complete grammar lattice; b) Grammar Boolean algebra

Theorem 6. *The system $\mathfrak{L} = \langle \mathcal{L}, \succcurlyeq \rangle$ together with the lub and glb operators computed by Algorithms 10 and 11 forms a complete lattice.*

Proof. Besides the property given in Lemma 11, *lub* and *glb* operators are computed w.r.t. (5.7) (see Figure 5.12(b)), such that we have $ts(\curlyvee_{G \in \mathcal{L}} G) = \bigcap_{G \in \mathcal{L}} ts(G) = ts(\top)$, $bs(\curlywedge_{G \in \mathcal{L}} G) = \bigcap_{G \in \mathcal{L}} bs(G) = bs(\bot)$, which gives the uniqueness of $\top$ and $\bot$ elements. □

Similar to the subsumption relation, $\succcurlyeq$, the *lub* $\curlyvee$ and *glb* $\curlywedge$ operators are semantic-based.

Theorem 7. *In the complete lattice $\mathfrak{L} = \langle \mathcal{L}, \succcurlyeq \rangle$, $\forall G_1, G_2 \in \mathcal{L}$ we have:*

$$\begin{aligned} &\mathbb{S}(G_1 \curlyvee G_2) \supseteq \mathbb{S}(G_1) \cup \mathbb{S}(G_2). \\ &\mathbb{S}(G_1 \curlywedge G_2) \subseteq \mathbb{S}(G_1) \cap \mathbb{S}(G_2) \end{aligned} \tag{5.8}$$

Proof. The proof is straightforward from Definition 27, Theorem 5 and Lemma 11. □

Thus, the grammar lattice $\mathcal{L}$ is semantic-based (see Figure 5.15(a)).

It is straightforward to prove that the complete lattice $\mathfrak{L} = \langle \mathcal{L}, \succcurlyeq \rangle$ has the following properties:

idempotency	$G \curlywedge G = G$	$G \curlyvee G = G$
commutativity	$G_a \curlywedge G_b = G_b \curlywedge G_a$	$G_a \curlyvee G_b = G_b \curlyvee G_a$
associativity	$(G_a \curlywedge G_b) \curlywedge G_c = G_a \curlywedge (G_b \curlywedge G_c)$	$(G_a \curlyvee G_b) \curlyvee G_c = G_a \curlyvee (G_b \curlyvee G_c)$
absorption	$G_a \curlywedge (G_a \curlyvee G_b) = G_a$	$G_a \curlyvee (G_a \curlywedge G_b) = G_a$
top	$\top \curlywedge G = G$	$\top \curlyvee G = \top$
bottom	$\bot \curlywedge G = \bot$	$\bot \curlyvee G = G$
interdefinability	$G_a \succcurlyeq G_b$ iff $G_a \curlywedge G_b = G_b$	$G_a \succcurlyeq G_b$ iff $G_a \curlyvee G_b = G_a$
distributivity	$G_a \curlywedge (G_b \curlyvee G_c) =$	$G_a \curlyvee (G_b \curlywedge G_c) =$
	$= (G_a \curlywedge G_b) \curlyvee (G_a \curlywedge G_c)$	$= (G_a \curlyvee G_b) \curlywedge (G_a \curlyvee G_c)$

In proving the above properties it is crucial that Procedure `Op` always computes the compositional constraints $\Phi_{\curlyvee}$ and $\Phi_{\curlywedge}$ at the end, and thus they are independent of the order in which the operators are applied in each of the equality sides.

5.3.1.3 Grammar Boolean Algebra

As is known, a (grammar) lattice $\mathfrak{L}_b = \langle \mathcal{L}_b, \succcurlyeq_\bot \rangle$ with $\top$ and $\bot$ elements, defines a *Boolean algebra* if $\forall G \in \mathcal{L}_b \;\; \exists \bar{G} \in \mathcal{L}_b$ uniquely determined[15] (called the *complement* of G), such that $G \curlyvee \bar{G} = \top$ and $G \curlywedge \bar{G} = \bot$.

In order for our grammar lattice to define a Boolean algebra, all pairs of elements, $G, \bar{G} \in \mathcal{L}_b$ must obey the following condition for all grammar rules (see Figure 5.16):

$$\begin{aligned} bd(G) \cup bd(\bar{G}) &= bd(\top) \cup bd(\bot) \\ bd(G) \cap bd(\bar{G}) &= \emptyset \end{aligned} \tag{5.9}$$

[15]The uniqueness of $\bar{G}$ is guaranteed in distributive lattices (Gierz et al., 2003).

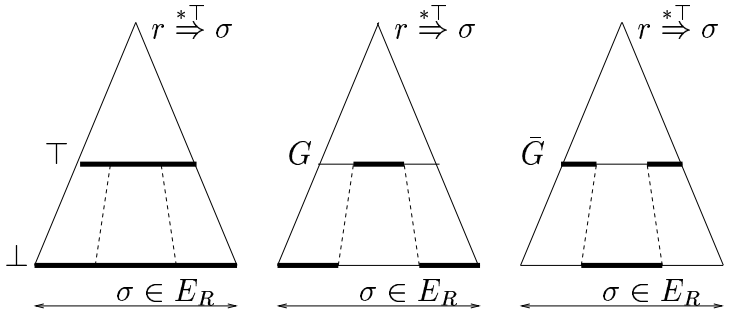


Figure 5.16: Boolean algebra

From the above relations it results that:

$$ts(G) \cap ts(\bar{G}) = ts(\top)$$
$$bs(G) \cap bs(\bar{G}) = bs(\bot) \tag{5.10}$$

for all grammar rules, and thus $G \curlyvee \bar{G} = \top$ and $G \curlywedge \bar{G} = \bot$ hold, i.e. the grammar lattice forms a Boolean algebra. This means that all rules of all the lattice's grammars contain only nonterminals from the corresponding rules of the $\top$ or $\bot$ grammars. For this, the rule derivation step must guarantee the complete derivation of each nonterminal down to the $\bot$ element level. Thus, we will have the following definitions regarding the rule derivation step:

Definition 29. *We call the rule bottom derivation step:*

$$\frac{A(\sigma_A) \rightarrow \alpha B(\sigma_B^*)\gamma : \Phi_A \qquad B(\sigma_B) \rightarrow \beta : \Phi_B}{A(\sigma_A) \rightarrow \alpha \beta_{\bot} \gamma : \Phi'_A} \tag{5.11}$$

E_R parsing preserving, and use the notation $r_A \overset{r_B}{\vdash}_{\bot} r'_A$, if $r_A \overset{*G}{\Rightarrow} \sigma_A \wedge r'_A \overset{*G'}{\Rightarrow} \sigma_A \wedge \sigma_A \in E_R \wedge r_B \overset{*G}{\Rightarrow} \sigma_B, \beta_{\bot} \overset{*\bot}{\Rightarrow} \sigma_B$.

Definition 30. *A grammar G' is one-step bottom derived from a grammar G, $G \overset{r_1}{\vdash}_{\bot} G'$, if $\exists r, r_1 \in P_G \wedge \exists r', r_1 \in P_{G'}$, s.t. $r \overset{r_1}{\vdash}_{\bot} r'$. A grammar G' is bottom derived from a grammar G, $G \overset{*}{\vdash}_{\bot} G'$, if it is obtained from G in n-derivation steps: $G \overset{r_1}{\vdash}_{\bot} \cdots \overset{r_n}{\vdash}_{\bot} G'$, where n is finite. We extend the notation so that we have $G \overset{*}{\vdash}_{\bot} G$.*

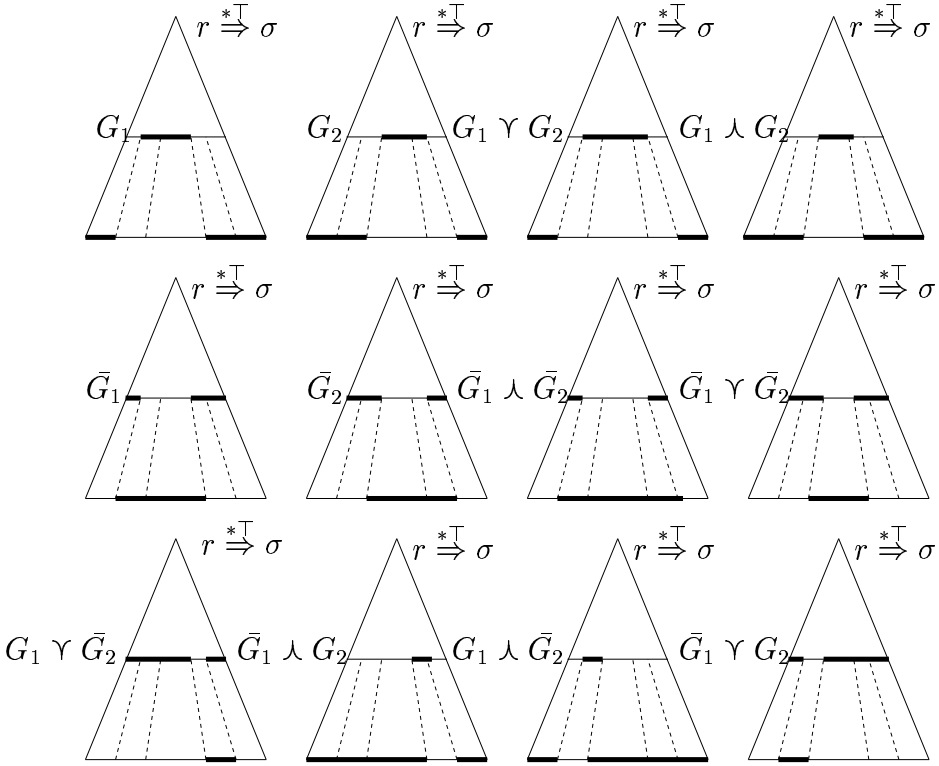


Figure 5.17: *lub* and *glb* operators in Boolean algebra

Definition 31. *If $G, G' \in \mathcal{L}_b$, we say that G bottom subsumes G', i.e., $G \succcurlyeq_{\perp} G'$, iff $G \overset{*}{\vdash}_{\perp} G'$.*

Given these bottom grammar derivation definitions the grammar lattice $\mathfrak{L}_b = \langle \mathcal{L}_b, \succcurlyeq_{\perp} \rangle$ obeys all the properties given in Section 5.3.1.2 and moreover, it forms a semantic-based Boolean algebra (see Figure 5.15(b)). In Figure 5.17 is given an example of *lub* and *glb* operator application, where grammar difference is $G_1 - G_2 = G_1 \curlywedge \bar{G}_2$, and where it can be seen that the de Morgan's laws also hold, i.e.:

$$\begin{aligned} \overline{G_1 \curlywedge G_2} &= \bar{G}_1 \curlyvee \bar{G}_2 \\ \overline{G_1 \curlyvee G_2} &= \bar{G}_1 \curlywedge \bar{G}_2 \end{aligned} \tag{5.12}$$

We notice that given the complete grammar lattice $\mathfrak{L} = \langle \mathcal{L}, \succcurlyeq \rangle$ (Theorem 7), we have that $\mathcal{L} \supseteq \mathcal{L}_b$ due to the fact that the rule bottom derivation entails the rule derivation, i.e. if $G \overset{*}{\vdash}_{\perp} G'$ then $G \overset{*}{\vdash} G'$. Thus the grammar Boolean algebra is a sublattice of the complete grammar lattice (see Figure 5.21(b)).

As an example, the grammars in Figure 5.10 form a boolean algebra, where $G_1 = \bar{G}_2$.

5.3.1.4 Learnability Theorem

In accordance with Assumption 3 (Section 5.1.1) the rules corresponding to the grammar preterminals (POS) are given (i.e. $T_G(\emptyset)$). Thus, for a given representative example set, E_R, we can construct the grammar $\perp$ using the robust parser ($P_{\perp} = Bottom(E_R)$). In order to build the $\top$ element, we need to apply the grammar generalization procedure starting from the $\perp$ element. This requires knowledge of the sublanguage E_σ since the partial order of the grammar lattice is semantic-based ($\mathbb{S}(G) = L_\sigma(G) \cap E_\sigma$).

The grammar generalization is determinate if the rule generalization step is determinate.

Definition 32. *A grammar rule $r'_A \in P_G$ is determinate generalizable by the rule generalization step* (5.3) *if $\exists \beta \in rhs(r'_A)$ and $\exists!\ r_B = B \rightarrow \beta$: Φ_B (i.e., one and only one rule r_B), s.t. $r'_A \overset{r_B}{\dashv} r_A$ with $\mathbb{S}(r'_A) \subset \mathbb{S}(r_A)$. We use the notation $r'_A \overset{1\subset}{\dashv} r_A$ for the determinate generalization step with semantic increase.*

In order to soundly build the $\top$ element of the grammar lattice, we must refine the definition of a grammar G *conformal* w.r.t. E_σ.

Definition 33 (Conformal, revised). *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 derivation step* (5.2) *is determinate ($\exists! r_B$) and guarantees the decrease in rule semantics ($\mathbb{S}(r_A) \supset \mathbb{S}(r'_A)$) for all grammars derived from G. We use the notation*

$r_A \overset{1\supset}{\vdash} r'_A$ *for the determinate derivation step with semantic decrease (rule* r_B *is unique and thus not specified).*

As a consequence, the only rule generalization steps (5.3) allowed in the grammar induction process are those which guarantee the same semantic relation $\mathbb{S}(r'_A) \subset \mathbb{S}(r_A)$,[16] which ensures that all the generalized grammars belong to the grammar lattice. We use the notation $r'_A \overset{r_B\subset}{\dashv} r_A$ for the generalization step with semantic increase (it can be nondeterminate, and thus r_B must be specified).

Let $chain_\top$ be a chain of rules in a LWFG $\top$ conformal w.r.t. a sublanguage E_σ, $chain_\top = \{B_k \to B_{k-1}\colon \Phi_{k\top}, \ldots, B_2 \to B_1\colon \Phi_{2\top}, B_1 \to \beta\colon \Phi_{1\top}\}$, such that $B_k \succ \cdots \succ B_2 \succ B_1$. All the chain rules, but the last, are unary branching rules. The last rule is the minimal chain rule.

The rules of a chain must ground derive equivalent representative syntagmas $\sigma_{B_k} \equiv \cdots \equiv \sigma_{B_1}$ (see Figure 5.18(a)), i.e., syntagmas that have the same string and the same semantic representation, but different categories.

For the $\perp$ grammar of a lattice that has $\top$ as its top element, the aforementioned chain becomes $chain_\perp = \{B_k \to \beta_\perp\colon \Phi_{k\perp}, \ldots, B_2 \to \beta_\perp\colon \Phi_{2\perp}, B_1 \to \beta_\perp\colon \Phi_{1\perp}\}$, where $\beta_\perp$ contains only preterminals and the rule order is unknown. By the parsing preserving property of the rule derivation step, the same equivalent representative syntagmas can be ground-derived from the $chain_\perp$ rules (see Figure 5.18(b)).

We denote by $chain = \{r_k, \ldots, r_2, r_1\}$, one or more chains in any lattice grammar, where the rule order is unknown. The minimal chain rules, $r_m = min(chain)$, can always be determined if $r_m \in chain$ s.t. $\forall r \in chain - \{r_m\} \wedge r_m \overset{r}{\dashv}$

[16] This property allows the grammar induction based only on positive examples.

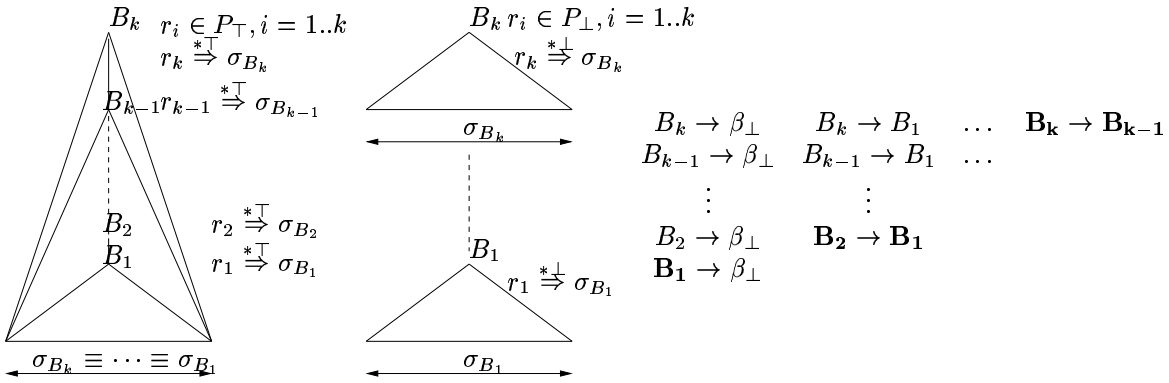


Figure 5.18: (a,b) Parsing trees for chain rules (in $chain_\top$ and $chain_\perp$, respectively); (c) The iterations of step 2 in Procedure `chains_recovery` ($chain_\top$ contains the diagonal rules)

r_{mg} we have that $\mathbb{S}(r_m) = \mathbb{S}(r_{mg})$. By the consequence of the conformal property, the generalization step $r_m \overset{r=}{\dashv} r_{mg}$ is not allowed, since it does not produce any increase in rule semantics. That is, a minimal chain rule cannot be generalized by any other chain rule, with an increase in its semantics.

Given $chain_\perp$ and the aforementioned property of the minimal chain rules, we can recover $chain_\top$ by Procedure `chains_recovery`.

Lemma 12. *Given a LWFG $\top$ conformal w.r.t. a sublanguage E_σ, for any grammar G derived from $\top$, all rules are determinate generalizable if all chains of the grammar $\top$ (i.e., all $chain_\top$) are known (e.g., recovered by Procedure* `chains_recovery`*).*

Proof. The only case of rule generalization step nondeterminism, with semantic increase, is introduced by the derivation of the unary branching rules of ordered $chain_\top$, which yields the unordered $chain_\perp$, where $B_i \to \beta_\perp \overset{B_j \to \beta_\perp \subset}{\dashv} B_i \to B_j$, holds for all $B_j \prec B_i$. Thus, keeping (or recovering) the ordered $chain_\top$ in any grammar G derived from $\top$, all the other grammar rules are determinate generalizable. We have three cases:

(i) $A \to \alpha\beta\gamma \colon \Phi'_A \overset{B \to \beta \colon \Phi_B \subset}{\dashv} A \to \alpha B\gamma \colon \Phi_A$.

(ii) $A \to \alpha B_i\gamma \colon \Phi'_A \overset{B_{i+1} \to B_i \colon \Phi_{B_{i+1}} \subset}{\dashv} A \to \alpha B_{i+1}\gamma \colon \Phi_A$.

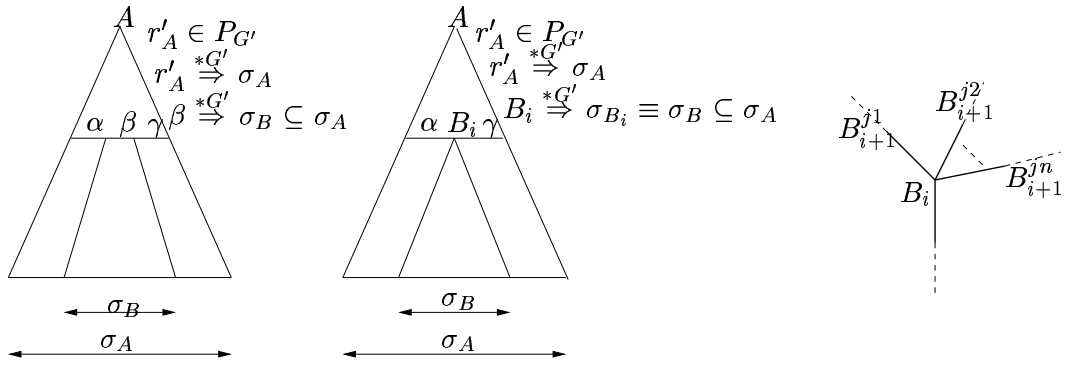


Figure 5.19: Determinate rule generalization: (a) no chains, (b) linear chains, (c) crossing chains

$$\text{(iii)}\ \ A \to \alpha B_i \gamma \colon \Phi'_A \quad \overset{B^j_{i+1} \to B_i \colon \Phi_{B^j_{i+1}} \subset}{\dashv} \quad A \to \alpha B^j_{i+1} \gamma \colon \Phi_A.$$

In the first case, (i), the nonterminal B either does not belong to any chain, or is the minimum in a chain. The uniqueness of the rule $B \to \beta$ is a consequence of the grammar unambiguity (see Figure 5.19(a)). In the second case, (ii), the nonterminal B_i belongs to a linear chain, and thus it occurs only once as a right-hand side of a unary branching rule, $B_{i+1} \to B_i$ (see Figure 5.19(b)). In the third case, (iii), the nonterminal B_i is a crosspoint of grammar chains, and thus it occurs as right-hand side of all unary branching rules $B^j_{i+1} \to B_i, j = 1..n$. In order to have only one such rule that allows the generalization step with semantic increase, the following condition must hold (see Figure 5.19(c)):

$$\mathbb{S}(B^{j1}_{i+1}) \bigcap_{\equiv} \mathbb{S}(B^{j2}_{i+1}) \equiv \mathbb{S}(B_i), \forall j1, j2 \in 1..n, j1 \neq j2 \tag{5.13}$$

□

Algorithm 13 builds the lattice $\top$ element, $\top \leftarrow Top(E_R, E_\sigma)$. In step 1, the Procedure `chains_recovery` detects all $chain = chain_\perp$, which contain rules with identical right-hand side. In step 2, all $chain_\perp$ rules are transformed in $chain_\top$ form, by generalizing them through the minimal chain rule (see also Figure 5.18(c)). The generalization step $r \overset{r_m \subset}{\dashv} r_g$ guarantees the semantic increase $\mathbb{S}(r_g) \supset \mathbb{S}(r)$ for

Algorithm 13: Top(E_R, E_σ)

$P_\perp \leftarrow Bottom(E_R)$
$P_\top \leftarrow$ `chains_recovery` $(P_\perp, E_R, E_\sigma)$ /*$P_\top$ is determinate generalizable */
while $\exists r \in P_\top$ *s.t.* $r \overset{1\subset}{\dashv} r_g$ **do**
  $r \leftarrow r_g$
return $P_\top$

all the rules r which are generalized through r_m, thus being the inverse of the rule derivation step in the grammar lattice. The rules r are either chain rules, or rules having the same left-hand side as the chain rules. The returned set $P_\perp$ contains all $chain_\top$ unary branching rules of the $\top$ grammar. Therefore, in Algorithm 13 the set $P_\top$ initially contains determinate generalizable rules, and the "while loop" can determinately generalize all the grammar rules.

Procedure `chains_recovery`$(P_\perp, E_R, E_\sigma)$

while $E_R \neq \emptyset$ **do**
1 $\sigma \leftarrow first(E_R)$
  $chain \leftarrow \{r \in P_\perp | r \overset{*\perp}{\Rightarrow} \sigma_r \wedge \sigma_r \equiv \sigma\}$ /* $chain = chain_\perp$ */
  $lhs\_chain \leftarrow \{lhs(r) | r \in chain\}$
  $E_R \leftarrow E_R - \{\sigma_r \in E_R | r \in chain \wedge r \overset{*\perp}{\Rightarrow} \sigma_r\}$
2 **while** $|chain| > 1$ **do**
    /* chain recovery */
    $r_m \leftarrow min(chain)$ /* r_m cannot be generalized with semantic increase */
    $chain \leftarrow chain - \{r_m\}$
    $lhs\_chain \leftarrow lhs\_chain - \{lhs(r_m)\}$
    foreach $r \in P_\perp \wedge lhs(r) \in lhs\_chain$ *s.t.* $r \overset{r_m\subset}{\dashv} r_g$ **do**
      $r \leftarrow r_g$
return $P_\perp$ /* The returned $P_\perp$ contains all $chain_\top$ */

An example showing the full trace of Procedure `chains_recovery` is given below.

E_R	$P_\perp$	*Generalized Rules at:*			
		Iteration 1	*Iteration 2*	*Iteration 3*	*Itereration 4*
he is	* $\mathbf{AV0 \rightarrow Pro, Aux}$				
is he	$AV0 \rightarrow Aux, Pro$				
he is	* $AV1 \rightarrow Pro, Aux$	* $\mathbf{AV1 \rightarrow AV0}$			
he is not	$AV1 \rightarrow Pro, Aux, Aux$	$AV1 \rightarrow AV0, Aux$			
he is	* $AV2 \rightarrow Pro, Aux$	* $AV2 \rightarrow AV0$	* $\mathbf{AV2 \rightarrow AV1}$		
he can be	$AV2 \rightarrow Pro, Aux, Aux$	$AV2 \rightarrow AV0, Aux$	$AV2 \rightarrow AV1, Aux$		
he is	* $AV3 \rightarrow Pro, Aux$	* $AV3 \rightarrow AV0$	* $AV3 \rightarrow AV1$	* $\mathbf{AV3 \rightarrow AV2}$	
he has been	$AV3 \rightarrow Pro, Aux, Aux$	$AV3 \rightarrow AV0, Aux$	$AV3 \rightarrow AV1, Aux$	$AV3 \rightarrow AV2, Aux$	
he is	* $AV4 \rightarrow Pro, Aux$	* $AV4 \rightarrow AV0$	* $AV4 \rightarrow AV1$	* $AV4 \rightarrow AV2$	* $AV4 \rightarrow AV3$
he is being	$AV4 \rightarrow Pro, Aux, Aux$	$AV4 \rightarrow AV0, Aux$	$AV4 \rightarrow AV1, Aux$	$AV4 \rightarrow AV2, Aux$	$AV4 \rightarrow AV3, Aux$

Table 5.2: A trace of Procedure `chains_recovery` for auxiliary verb constructions with 4 iterations. Chain rules are marked with *, and the minimal chain rules are in bold.

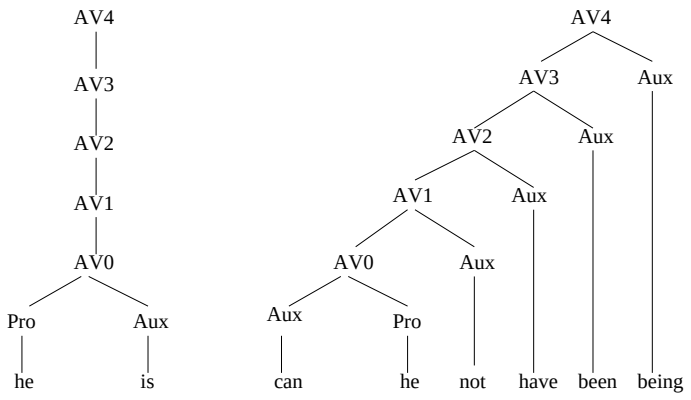
Example. Table 5.2 shows the trace of Procedure `chains_recovery` for auxiliary verb constructions. It can be noticed that at the end, all the rules are determinate generalizable. For simplicity, only the strings corresponding to E_R are shown (i.e., their semantic molecules are not given). The constraints Φ, which are not shown, are computed at each step for the generalized rules. The sublanguage E_σ used for the generalization process is not given. As a note, the *Pro* nonterminal will be further generalized to *Sbj*. Figure 5.20 shows examples of parse trees corresponding to the ground derivation in the grammar returned by Procedure `chains_recovery`.

Theorem 8 (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 the procedure Top(E_R, E_σ) computes the lattice $\top$ element such that $\top = G$.*

Proof. Since G is normalized (Definition 25), none of its rule can be generalized with increase in semantics. Starting with the $\bot$ element, after applying Procedure `chains_recovery`, all rules that can be generalized, with increase in semantics, through the rule generalization step (5.3), are determinate generalizable (Lemma 12, Definition 32, 33) . Since the rule generalization step and the rule derivation step are the inverse of each other, the process of grammar generalization from $\bot$ to $\top$ is the inverse of the derivation process from G to $\bot$, which is finite. This means that regardless of the grammar sequence $\bot, G_1, \ldots, G_n, \top$, there is a derivation process that yields the inverse sequence $G, G_n, \ldots, G_1, \bot$. Since $\mathbb{S}(G) = E_\sigma$ and $\mathbb{S}(G_i)$ are increasing (because G is conformal to E_σ), the generalization process ends at the semantic limit $\mathbb{S}(\top) = E_\sigma$, and thus $\top = G$. □

If the hypothesis of Theorem 8 holds, then any grammar induction algorithm that uses the complete lattice search space can converge to the lattice top element, using different search strategies.

We saw in Section 5.1.2 that if the representative examples are ordered, we have a more efficient algorithm (Algorithm 7) for learning the $\top$ element, for which we proved the theorem of induction. In this algorithm, the grammar learning is done bottom-up starting from the $\bot$ grammar, such that each learned rule belongs to the $\top$ grammar. This way, the generalization process is the inverse of the bottom derivation, and the search space is the one of a Boolean algebra since at each point in time, the grammar rules either belong to the $\top$ element or to the $\bot$ element. That



(a) (b)

Figure 5.20: (a) Chain rule deriving a syntagma from E_R; (b) Non-chain rule deriving a syntagma from E_σ

is, a grammar G in the Boolean algebra search space has the first rules (already learned) in the $\top$ element, while the others (not learned) in the $\bot$ element. The Boolean algebra-type search space gives us the algorithm efficiency (i.e, the $\top$ element is quickly obtained).

Also, when the grammar contains chains and the ordering relation among the chain nonterminals as well as the order of representative examples are unknown, the grammar can be iteratively learned based on the chain maximum semantics property (Algorithm 9). Moreover, since the search space is a complete grammar lattice based on monotone semantics, the issue of generalization step nondeterminism is overcome.

For both Theorem 2 and Theorem 3, the given normalized grammar G (i.e., its rules cannot be further generalized) and the given sublanguage E_σ, uniquely determines the set of representative examples E_R and the grammar complete lattice, as well as the grammar Boolean algebra as its sublattice. The grammar G is the

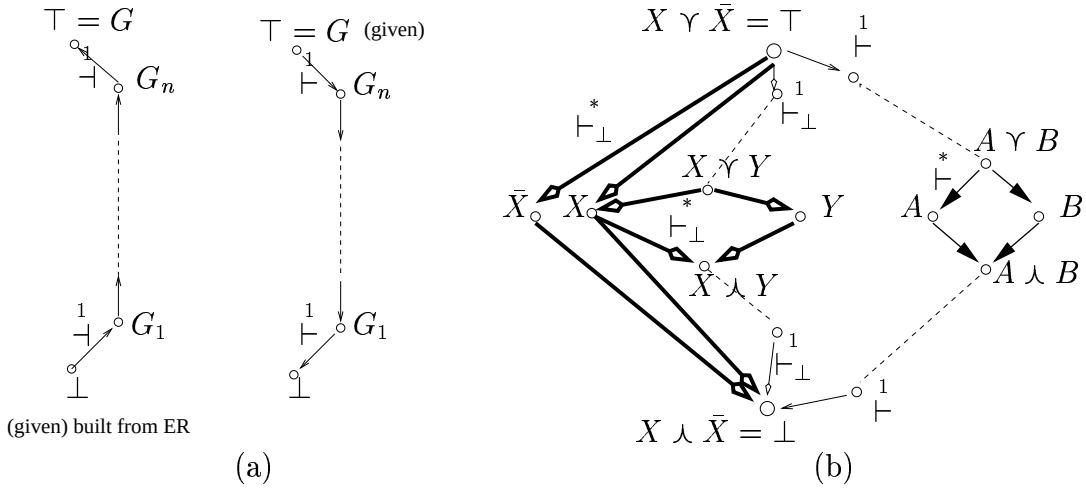


Figure 5.21: Complete Grammar Lattice. a) $\top$ built based on the determinate rule generalization step $\overset{1}{\dashv}$; b) grammar Boolean algebra based on bottom rule derivation $\vdash_\bot$ is a sublattice in the complete grammar lattice based on rule derivation $\vdash$.

lattice $\top$ element. The learning is done in the grammar lattice search space, and the learned grammar has the property that its rules cannot be further generalized. This means that the learned grammar is the lattice $\top$ element which is unique, and thus both theorems are proved.

Regarding grammar Boolean algebra, we notice that these algebras can be used for individual learning of grammars that have disjoint sets of nonterminals. The efficiency of merging these grammars can be increased using the *lub* operator, instead of relearning the merged grammar (see Figure 5.15(b) and 5.21(b)).

5.3.2 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. Then, in Section 5.3.2.2 we prove that our Lexicalized Well-Founded Grammars are de-

cidable. We conclude this section by presenting our new relational learning model for LWFG induction, i.e., the *GARS* model.

5.3.2.1 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) describe formally the ILP-problem and relate it to Gold (1967) and PAC-learnability (Valiant, 1984) frameworks. They show that ILP-problem and Gold's identification in the limit are not strongly connected, but they define PAC-learnability for the ILP setting. We briefly present the ILP problem and then discuss how our grammar induction problem is framed in this approach.

ILP Learning Problem [17]

Given:

- a correct provability relation $\vdash$ for a first-order language L, i.e., for all $A, B \in L$: if $A \vdash B$, then $A \models B$,
- background knowledge B in language LB: $B \in LB \subseteq L$,
- positive and negative examples $E = E^+ \cup E^-$ in language $LE \subseteq L$ consistent with B $(B, E \not\vdash \square)$ and not a consequence of B $(\forall e \in E\colon B \not\vdash e)$, and
- hypothesis language $LH \subseteq L$.

Find a hypothesis $H \in LH$ such that:

(i) $(B, H, E \not\vdash \square)$, i.e., H is consistent with B and E,

[17] Taken from (Kietz and Džeroski, 1994).

(ii) $(B, H \vdash E^+)$, i.e., $H \wedge B$ explain E^+, and

(iii) $(\forall e \in E^- : B, H \nvdash e)$, i.e., $H \wedge B$ do not explain E^-.

The tuple $\langle \vdash, LB, LE, LH \rangle$ is called the *ILP-learning problem.* Deciding whether there exists such an $H \in LH$ is called the *ILP-consistency problem.* An algorithm which accepts any $B \in LB$ and $E \subseteq LE$ as input and computes such an $H \in LH$ if it exists, or "no" if it does not exist is called an *ILP-algorithm.*

Kietz and Džeroski showed that in order to ensure polynomial PAC-learnability we need to have polynomial solvability of the ILP-learning problem and to show that any good hypothesis is found from only a polynomial number of examples. But the general ILP-problem $\langle \models$, *Horn clauses, ground atoms, Horn clauses* $\rangle$ is undecidable. Thus, it is neither consistently identifiable in the limit, nor PAC-learnable. The question is what subclasses of first-order logic are efficiently learnable. Possible choices to restrict the ILP-problem are: the provability relation, $\vdash$ (also called the generalization model), 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.

Grammar Induction Problem in ILP-setting

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

- The provability relation, $\vdash$, is given by the robust parsing, and we denote it by $\vdash_{rp}$. We use the "parsing as deduction" technique (Shieber, Schabes, and

Pereira, 1995) (see Section 3.5.1). Using this technique, 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 (a type of DCG 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 sytagmas 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.

That is, our grammar induction problem can be formulated as $\langle \vdash_{rp}$, *LWFG rules (type of DCG rules) + elementary syntagmas (ground atoms), representative sublanguage(ground atoms), LWFG rules (type of DCG rules)* $\rangle$.

5.3.2.2 Decidability of LWFGs

In this section we show that LWFGs are decidable, i.e., we show that given a LWFG G and any syntagma σ, there is an algorithm that can decide in a finite number of steps whether $\sigma \in L_\sigma(G)$. This is equivalent to say that the grammar language can be recognized by a Turing machine that halts on all inputs.

As given in Chapter 3, $L_\sigma(G) = \{\sigma | \sigma = (w, w'), w \in \Sigma^+, \exists A \in N_G, A \stackrel{*}{\Rightarrow} \sigma\}$. We showed that the ground derivation is equivalent to parsing provability, i.e, $A \stackrel{*}{\Rightarrow}$

σ iff $G \vdash_{rp} A(\sigma)$. Thus for a LWFG G and any syntagma σ we have that:

- $\sigma \in L_\sigma(G)$ if $G \vdash_{rp} A(\sigma)$
- $\sigma \notin L_\sigma(G)$ if $G \nvdash_{rp} A(\sigma)$

Thus the decidability is reduced to proving the termination of parsing. A rule $A(\sigma) \rightarrow B_1(\sigma_1), \ldots, B_n(\sigma_n) \colon \Phi(\bar{\sigma})$ in a LWFG G ensures that:

- $\sigma > \sigma_i, 1 \leq i \leq n$. We have two cases: a) if $n > 1$ then $|w| > |w_i|$ which holds true since the empty string cannot be derived; and b) if $n = 1$, which corresponds to unary branching rules $A(\sigma) \rightarrow B_1(\sigma_1) \colon \Phi(\bar{\sigma})$, we have that $w = w_1$ and $A \succeq B$ which holds true because there are only chains where the nonterminals are ordered, and there are no cycles.
- the solving of the $\Phi(\bar{\sigma})$ constraints, which are path equations, is guaranteed to terminate as it results from the properties given in Chapter 4. The interpretation, Φ_{onto}, guarantees termination because the semantic interpretation is performed in the framework of natural language as problem formulation (i.e., without reasoning).

From these two properties it results the termination of the parsing algorithm.

5.3.2.3 GARS 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 ILP-learning setting outlined above, such that G is unique and $E_\sigma \subseteq L_\sigma(G)$.

The representative sublanguage defines the reduced semantics that is used as the performance criterion for grammar rule generalization during grammar learning. LWFGs are decidable and thus the ILP learning problem is decidable. 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. This class is a class of constraint-based grammars which capture syntax and semantics (LWFG) and which are learnable by relational learning methods.

Practically, the GARS model uses two algorithms for LWFG learning. The first algorithm learns from an ordered set of representative examples and has as hypothesis search space a Boolean algebra, which guarantees increased learning efficiency.

The second algorithm learns from unordered representative examples and has as hypothesis search space a complete grammar lattice, which has as a sublattice the Boolean algebra of the first algorithm. Even if this algorithm is less efficient — it requires several iteration steps to converge to the same grammar as the first

algorithm — its importance is twofold:

- we do not always know the right order of examples, so learning from unordered examples is a practical need.
- this iterative algorithm allows a sound grammar revision.

The second property can be used to revise the grammar only by revising the set of representative examples and elementary semantic molecules at the lexicon level. In other words, the burden of rewriting by hand the grammar rules and the compositional constraints is replaced by a much easier task of revising the examples. This sound grammar revision makes the GARS model suitable for incrementally increasing the coverage of the learned grammar. In this way, GARS can be seen as a learning model suitable for approximating natural language coverage. The incremental learning with revision is illustrated in Chapter 7, where we describe our learning system, which implements all the theoretical algorithms presented in Part I (Chapters 3, 4, 5) for learning an experimental LWFG. Samples of the results are also given in Appendix A.

Before presenting our LWFG learning system and the learned experimental grammar, we present in Chapter 6 the expressiveness of our semantic representation OntoSeR for all the linguistic phenomena covered by this learned grammar. This chapter is one of the qualitative evaluations of our theoretical GARS model.

Part II

Applicative Part

Chapter 6

Expressiveness of the Semantic Representation (OntoSeR)

The general problem of text interpretation involves the determination of the semantic relations among entities and the events they participate in, answering to questions, such as "who" did "what" to "whom", "when", "where", "how" and "why". In this dissertation, we take the meaning to be the set of answers given to all the questions w.r.t. an utterance or set of utterances. In Section 4.2 we formally defined the meaning of a syntagma, M_σ, as the set of all query-answers that can be derived by adding the assertional logical form K_σ to the system's prior knowledge (K_o,K_i,K_d) (see (4.20), pg. 88).

In this chapter, we are presenting in detail the expressiveness of OntoSeR from a linguistic point of view, its usefulness for a straightforward mapping from utterance representation to knowledge representation (ontology level), as well as it adequacy for obtaining the meaning of utterances. Therefore, this chapter will offer a *qualitative evaluation* of the learning grammar method developed in this dissertation, by using an experimental learned grammar. The focus is not the learned grammar in itself, which can be at any moment revised, but precisely the

$$
\begin{aligned}
\langle \text{OntoSeR} \rangle &\stackrel{\text{def}}{=} \langle \text{AP} \rangle \mid \langle \text{OntoSeR} \rangle \langle \text{lop} \rangle \langle \text{OntoSeR} \rangle \\
\langle \text{AP} \rangle &\stackrel{\text{def}}{=} \langle \text{conceptID} \rangle \ . \ \langle \text{attr} \rangle = \langle \text{concept} \rangle \\
\langle \text{AP} \rangle &\stackrel{\text{def}}{=} \langle \text{conceptID} \rangle = \langle \text{conceptID} \rangle \langle \text{coord} \rangle \langle \text{conceptID} \rangle \\
\langle \text{concept} \rangle &\stackrel{\text{def}}{=} \langle \text{conceptID} \rangle \mid \langle \text{conceptName} \rangle \\
\langle \text{conceptID} \rangle &\stackrel{\text{def}}{=} \langle \text{logicalVariable} \rangle \\
\langle \text{conceptName} \rangle &\stackrel{\text{def}}{=} \langle \text{lexicalWord} \rangle \\
\langle \text{attr} \rangle &\stackrel{\text{def}}{=} \langle \text{attrID} \rangle \mid \langle \text{attrName} \rangle \\
\langle \text{attrID} \rangle &\stackrel{\text{def}}{=} \langle \text{logicalVariable} \rangle \\
\langle \text{attrName} \rangle &\stackrel{\text{def}}{=} \langle \text{lexicalWord} \rangle \\
\langle \text{coord} \rangle &\stackrel{\text{def}}{=} \langle \text{lexicalCoord} \rangle \\
\langle \text{lop} \rangle &\stackrel{\text{def}}{=} \wedge
\end{aligned}
$$

Figure 6.1: Definition of OntoSeR (syntactic definition)

usefulness of the semantic representation for covering diverse linguistic phenomena.

First, we need to refine the definition of OntoSeR given in Section 3.2.1. The new definition is shown in Figure 6.1. As can be seen from this definition, the variables in our representation are either concept IDs or attribute IDs in the ontology; hence, we can argue that OntoSeR is an ontology query language. The logical operator lop, which we consider in this dissertation is the logical conjunction ($\wedge$).[1] The coord operator is one of the linguistic coordinators, such as *and*, *or*, *but* (see details in Section 6.1.5).

As we explained in Section 4.2, we took into account several levels of representation in order to get from the utterance representation to the knowledge rep-

[1] If the grammar would cover other constructions, such as "if then else" constructions, other logical operators would be required.

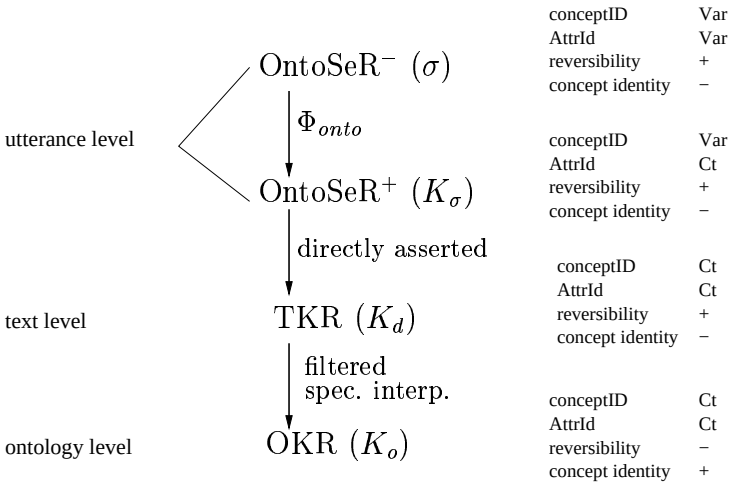


Figure 6.2: Levels of representation

resentation. We mainly have three levels: utterance level, text level and ontology level (see Figure 6.2).

At the *utterance level*, we call **OntoSeR**$^-$ the semantic representation that corresponds directly to a syntagma σ, before the ontology constraint Φ_{onto} is applied. Both the conceptIDs and attrIDs remain variables. After the application of Φ_{onto} during parsing (i.e., a semantic interpretation, which can be weak, such as role compatibility and consistency check), the assertional form K_σ of the syntagma σ is obtained. We call this form **OntoSeR**$^+$. At this level, the attrIDs become constant, while the conceptIDs remain variables to allow further composition to take place (we are still at the utterance level). Both at the OntoSeR$^-$ and OntoSeR$^+$ levels, we can exploit the reversibility of the grammar, since both these representations are used during parsing/generation. Reversibility signals the nondeterminism at the parsing/generation level.

The *text level* or discourse level representation, **TKR**, represents the asserted

representations (K_d). The conceptIDs become constants, and no composition can happen at this level. However, we still have (indirect) reversibility, since TKR represents all the asserted OntoSeRs$^+$. Therefore, all the information needed for reversibility is still present.

The knowledge representation at the ontology level, **OKR** (K_o), is obtained after filtering and task-specific interpretation. For example, we filter determiners, and some verb forms, such as aspect. In this dissertation, task-specific interpretation is geared mainly towards terminological interpretation. OKR is a directed acyclic graph (DAG) $G = (V, E)$, where vertices V are concepts (corresponding to nouns, verbs, adjectives, adverbs, pronouns, cf. Quine's criterion (Sowa, 1999, page 496)), while edges are semantic roles given by verbs, prepositions, adjectives and adverbs (see more details in Chapter 8). We can consider the nodes of the graph as frames in an ontology and the edges as the slots of the frames. At the OKR level we assume the **principle of concept identity** which means that there is a bijection between a vertex in OKR and a referent. For example, if we do not have pronoun resolution, the pronoun and the noun it refers to will be represented as two separate vertices in the graph. Both the concepts (vertices) and the semantic roles (edges) form hierarchies of concepts and semantic roles, respectively. Based on these hierarchies we can model the semantic context for the interpretation that takes place both locally, from OntoSeR$^-$ to OntoSeR$^+$ (i.e., Φ_{onto}), and globally from TKR to OKR (where additional discourse context can be used). In this dissertation, we consider only a fairly weak semantic context given by the admissibility relations that we can find at the level of lexical entries. For the verb thematic roles we considered the thematic roles derived from Dorr's "LCS Database" (Dorr, 1997). For

adjectives, and adverbs we took the roles (properties) from WordNet (Miller, 1990), and for those we did not find in WordNet we looked at FrameNet (Baker, Fillmore, and Lowe, 1998), or we manually added them. For prepositions we considered the "LCS Database", or we have manually added specific semantic roles.

Since text meaning is defined as being all the answers given to all questions w.r.t. that text (see Section 4.2), the meaning is implicitly contextual. It indeed depends on all the filtering and consistency checks that take place at various levels of representation.

An answer is a vertex in the OKR of an utterance, together with all the edges incident from/to it. A question is a subgraph of the utterance graph where the wh-word substitutes the answer concept (see Figure 6.5).

In the remainder of this introductory part we will consider some examples to illustrate all the concepts introduced until now.

Example In Figure 6.3 we show all the levels of representation starting from OntoSeR$^+$ for the utterance *John has been loving Mary.* We can see that the only difference between OntoSeR$^+$ and TKR is that the conceptID are variables in OntoSeR$^+$ (A,B,C), while they are constants in TKR ($\sim$1,$\sim$2,$\sim$3). At both these levels we have all the semantic information that appears in the utterance (verb tense, aspect, etc). At the OKR level, some filtering can take place: for example we can ignore verb aspect (i.e., the progressive pg, and perfect pf forms). Moreover, all concepts are denoted by # (frame identifiers). In Figure 6.3 we can see both the OKR and a graph representation that is its equivalent. The graphical representation displays clearly the directed graph where the concepts are vertices, and semantic roles are edges. We can also notice that some vertices are not concepts

OntoSeR$^+$

```
[A.name='John',B.tense=pr,B.pf=y,B.pg=y,B.is_a=love,B.exp=A,B.perc=C,
 C.name='Mary']
```

TKR

```
~1.name='John'
~2.tense=pr
~2.pf=y
~2.pg=y
~2.is_a=love
~2.exp=~1
~2.perc=~3
~3.name='Mary'
```

OKR

```
#love2.tense=pr
#love2.exp=#'John'
#love2.perc=#'Mary'
```

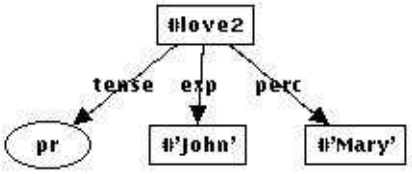


Figure 6.3: Representations of *John has been loving Mary*

but simply values of slots (e.g., tense and aspect, modals). For readability reasons, in the remainder of this chapter we generally keep only the graphical representation of the OKR.

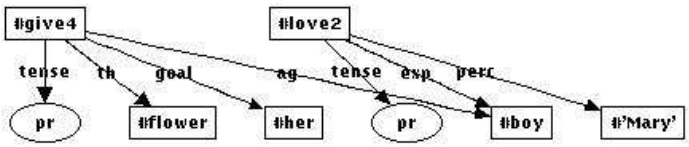
In Figure 6.4 we want to emphasize briefly the principle of concept identity. In 6.4(a) we show the TKR and OKR of the utterance *The boy who loves Mary gives her a flower*. Since we do not deal with anaphora resolution, we have two vertices in the graph: one corresponding to *Mary*, the other to *her*. Another point that we want to make is that the concept identity principle allows us to have an implicit merging procedure at the OKR level. In 6.4(b) we have the TKR and OKR representation of the two utterances *The boy loves Mary* and *The boy gives her a flower*. At the TKR level we have *the boy* corresponding to two concepts ID ($\sim$1, and $\sim$4, respectively). However, at the OKR level they become the same #boy

TKR

```
~1.det=the
~1.is_a=boy
~1.is_a=who
~2.tense=pr
~2.is_a=love
~2.exp=~1
~2.perc=~3
~3.name='Mary'
~4.tense=pr
~4.is_a=give
~4.ag=~1
~4.th=~5
~4.goal=~6
~6.is_a=her
~5.det=a
~5.is_a=flower
```

OKR

```
#love2.tense=pr
#love2.exp=#boy
#love2.perc=#'Mary'
#give4.tense=pr
#give4.ag=#boy
#give4.th=#flower
#give4.goal=#her
```



(a) *The boy who loves Mary gives her a flower*

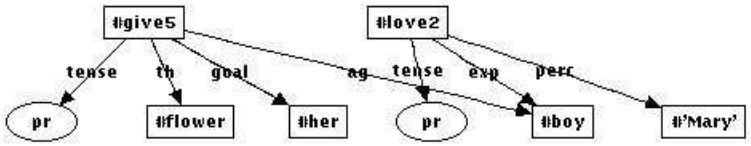
TKR 1+2

```
~1.det=the
~1.is_a=boy
~2.tense=pr
~2.is_a=love
~2.exp=~1
~2.perc=~3
~3.name='Mary'

~4.det=the
~4.is_a=boy
~5.tense=pr
~5.is_a=give
~5.ag=~4
~5.th=~6
~5.goal=~7
~7.is_a=her
~6.det=a
~6.is_a=flower
```

OKR 1+2

```
#love2.tense=pr
#love2.exp=#boy
#love2.perc=#'Mary'
#give5.tense=pr
#give5.ag=#boy
#give5.th=#flower
#give5.goal=#her
```



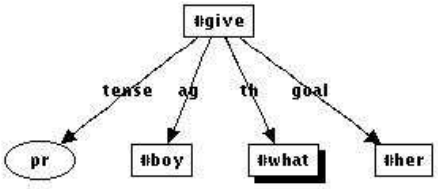
(b) *1. The boy loves Mary 2. The boy gives her a flower*

Figure 6.4: TKR/OKR examples

Question: What does the boy give her?

Answer: #flower

OKR



OKR

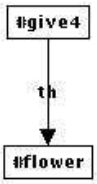


Figure 6.5: Question and answers for utterance(s) in Figure 6.4(a)

(the determiners are filtered). This way, merging these two utterances we obtain the same OKR as the OKR of the utterance given in 6.4(a).

As previously mentioned, the OKR representation allows us to obtain the meaning of a text through answers to questions. For the sake of readability, for each utterance presented in this chapter, we choose to show only one or two questions together with their answers. This will be eloquent to illustrate our point. In Figure 6.5 we present the question *What does the boy give her?* asked in conjunction to either the utterance in Figure 6.4(a) or the merged utterances in Figure 6.4(b), along with the answer that we obtain. As we can see, the OKR of the question is a subgraph of the utterance(s) graph where the wh-word is the vertex corresponding to the answer vertex (in this case #what matches the concept #flower). The answer is the vertex representing the concept #flower. At the rendering of OKR we can choose to give a more detailed context of the vertex, for example all the edges incident from/to it (see Figure 6.5 where the OKR of the answer shows also the context of the concept vertex). In Chapter 8, we will present examples where the answer concept is given in association with its subgraph (that represents nodes which determine the concept).

In the remainder of this chapter, we will present the expressiveness only at

the TKR and OKR level. All the examples presented are covered by our learned grammar (described in Chapter 7). In Section 6.1 we present the representation of lexical items (verbs, nouns, adjective, adverbs, prepositions and coordinators). Section 6.2 presents the treatment of raising and control constructions. In Section 6.3 we describe the treatment of relative clauses and wh-questions, showing how filler-gap dependencies are handled in our framework. In Section 6.4 we discuss the issue of ambiguity.

6.1 The Lexicon and Elementary Semantic Molecules

In the Lexicalized Well-Founded Grammar formalism, an elementary semantic molecule corresponds to each terminal symbol (word, or lexical item). In order to establish this correspondence, we consider a set of elementary semantic molecule templates that correspond to lexical categories. Within the learning framework, the lexicon and these templates are *a priori* given as background knowledge. Table 6.1 gives all lexical categories (parts of speech), the number of their associated elementary semantic molecule templates and the number of attributes that appear in the molecule heads. We should remind our reader that the attributes that appear in semantic molecule heads are of three types: feature, variable and constant (see Section 4.1).We want to point out that these elementary semantic molecule templates can be easily refined, and the grammar learning framework provides an implicit grammar revision. Thus, in general, different attributes or representations can be tried and the effort is minimal since it will not involve refining by hand the grammar rules or the constraints.

In the next sections we will briefly describe each of these lexical categories

No. of Lexical Categories	No. of Sem. Molecule Templates	No. of Attributes		
		Feature	Variable	Constant
aux	10	10	1	1
verb	22	12	5	1
adj	1	0	2	1
noun	3	8	4	1
pn	1	8	1	1
pro	1	8	1	1
apro	1	2	2	1
relpro	2	6	2	1
adv	1	0	1	1
det	1	0	1	1
prep	1	0	3	1
iprep	1	1	2	1
coord	1	0	3	1

Table 6.1: Lexical categories: Statistics

and the elementary semantic molecule templates associated with them.

6.1.1 Verbs

Verbs are central categories of any approach to syntax and semantics. In this dissertation, we give an account of representing complex verbal constructions, including tense, aspect, negation, modals, finite and non-finite verbs. For this, we introduce 32 semantic molecules templates (22 for main verbs and 10 for auxiliaries) and 12 features. As can be seen from Table 6.1, elementary molecules templates related to verbal constructions represent almost 70% of all the elementary semantic molecule templates. However, the semantic interpretation will be partial, depending on the specific application (e.g., a partial account for negation and modals for terminology, and tense and negation for some factual constructions). While the full semantic interpretation of complex verbal constructions is outside the scope of this disser-

$$(\text{gives})_I = \left(\begin{array}{l} \left[\begin{array}{ll} \text{cat} & \text{verb} \\ \text{vtype}^* & \text{norm} \\ \text{vft} & \text{fin} \\ \text{val} & \text{dv} \\ \text{vf} & \text{no} \\ \text{voice} & \text{act} \\ \text{tense}^* & \text{pr} \\ \text{pers}^* & \text{3rd} \\ \text{pg} & \text{no} \\ \text{pf} & \text{no} \\ \ldots & \\ \text{head} & X \\ \text{headS} & Y \\ \text{headC1} & Z1 \\ \text{headC2} & Z2 \end{array} \right]_h \\ \left\langle X.\text{tense} = \text{pr},\ X\text{is\_a=give},\ X.\text{Arg0}=Y,\ X.\text{Arg1} = Z1,\ X.\text{Arg2}=Z2 \right\rangle_b \end{array} \right)$$

Figure 6.6: Elementary semantic molecule for *gives*

tation, we provide an expressive representation that could be used to develop a semantic interpretation module for this purpose.

An example of an elementary semantic molecule for finite, ditransitive, active voice main verbs is given in Figure 6.6, with an instantiation shown for the lexical item *gives*.[2] The **head** of the elementary semantic molecule given in Figure 6.6 contains: *category attributes* (cat), *feature attributes* (vtype, vft, val, vf, voice, tense, pers, pg, pf), *variable attributes* (head, headS, headC1, headC2).

For lexical items, the *category attribute* specifies the part of speech, in this case, verb.

The *feature attributes* can be divided in two categories, based on their values in the elementary semantic molecule templates. The values of the attributes marked

[2]In the lexicon we have five forms of verbs associated with five types of elementary semantic molecules. Considering finite/nonfinite forms, and intransitive/transitive/ditransitive verbs, there are 22 distinct elementary semantic molecule templates.

by $*$ (tense, pers, vtype) are variables that get instantiated only when applied to a particular lexical item. This means that we do not require separate elementary molecule templates for different values of tense attribute, pers attribute, or vtype attribute (this attribute distinguishes between normal, raising or control verbs).[3] The rest of the feature attributes influence the verb semantics. For example, the valence attribute val determines the number of arguments the verb has. For ditransitive verbs, we have three arguments, as can be seen in Figure 6.6. The verb form type attributes vft specifies if the verb is finite or non-finite, which leads to different semantic representations. The attributes pf, pg specifies the perfect or progressive aspect of finite verbs, which is marked in the semantic representation.

In general, *variable attributes* are involved in the composition process and link a semantic molecule with other semantic molecules. For verbal constructions, the number of variable attributes is influenced by the value of val attribute. In the case of ditransitive verbs, we have four attributes: head which is the index of the verb, headS, which will be linked to the subject of the verb, and headC1, headC2, which will be linked to the two complements, respectively. The variables associated with this attributes are also present in the semantic representation.

The **body** of the elementary semantic molecule given in Figure 6.6 represents the semantic representation of the lexical item *gives* (i.e., its OntoSeR$^-$). The variables in the representation are concept or attribute/role identifiers in the ontology. Thus, X will map onto the concept associated with the verb *give*, while the arguments $Arg0$, $Arg1$,$Arg2$ will map onto the verb thematic roles (e.g., ag, th,

[3]Unlike other lexicalized approaches, such as HPSG, we do not use a different lexical structure for raising and control verbs, we only specify their vtype. These constructions are treated at the grammar rule level, as can be seen in Section 6.2.

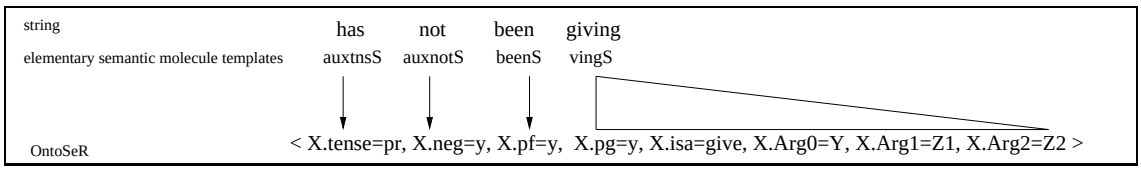


Figure 6.7: OntoSeR of a verbal construction with auxiliaries

goal). The variables Y, $Z1$, $Z2$ will map onto other concepts in the ontology that represent the arguments of the verb. As can be noticed, we work with a frame-like representation. In this way, the tense is actually a slot in the frame of the verb. Similarly, aspect, negation, modals are represented by slots in the verb's frame. We present in Figure 6.7 the semantic representation of the construction *has not been giving* and show schematically how each lexical element (more precisely, its elementary semantic molecule) contributes to the whole representation.

The composition is given by grammar rules that are learned from examples. A discussion of the total number of grammar rules learned for different phenomena is presented in Chapter 7. As we mentioned in the beginning of this section, in order to represent complex verbal constructions we introduced 32 elementary semantic molecule templates and 12 feature attributes. The main goal was to encode in the semantic representation (OntoSeR) all the information needed for the semantic interpretation, even if in this dissertation we only partially develop the interpretation module for verbal constructions.

Let us take two examples of complex verbal constructions in (1):

(1) a. Kim has not been reading the book.

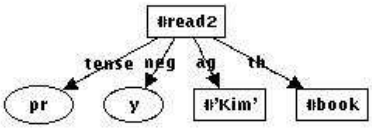
b. Kim could not have written the book.

Kim has not been reading the book.

TKR

```
~1.name='Kim'
~2.tense=pr
~2.neg=y
~2.pf=y
~2.pg=y
~2.is_a=read
~2.ag=~1
~2.th=~3
~3.det=the
~3.is_a=book
```

OKR

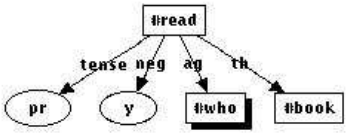


(a)

Questions

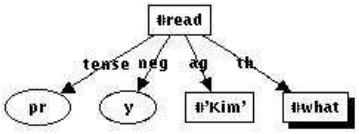
1. Who has not been reading the book?
2. Who doesn't read the book?

OKR



3. What has not been being read by Kim?

OKR

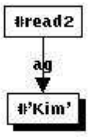


(b)

Answer for 1,2

```
[#'Kim']
```

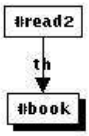
OKR



Answer for 3

```
[#book]
```

OKR



(c)

Figure 6.8: Representation and meaning for (1a)

In Figure 6.8 we show the text level representation (TKR) and the ontology-level representation (OKR), as well as three questions together with their answers for the construction (1a). From the TKR representation we can see that the verb's frame contains, besides its arguments, information about negation, tense, aspect (perfect and progressive). Thus, at this level, all the information pertaining to the utterance is kept in the representation. This allows us to obtain the utterance directly from the representation using the robust parser/generator (see Appendix A.3).

At the OKR level, filtering can take place. If we choose to ignore the aspect information (pf, pg) we obtain the representation given in Figure 6.8(a). The fact that the aspect is ignored from the meaning of the utterance, becomes clear when

we ask the questions: *Who has not been reading the book?* and *Who doesn't read the book?*. These two questions have the same OKR representation and thus give the same answer (shown in Figure 6.8). We can also ask questions at the passive voice: *What has not been being read by Kim?* and get an answer even if the utterance was in active voice. This shows that semantically, active and passive voice are equivalent (see Section 6.2.2.3 for other examples of active and passive constructions). However, if we ask the questions: *Who has been reading the book?*, or *Who had not been reading the book?* (i.e., no negation, and wrong tense) we fail to obtain an answer, since negation and tense are part of the utterance meaning.

A similar analysis is shown in Figure 6.9 for the utterance in (1b). We show how modals are represented and illustrate that we can choose to represent modals and negation at the OKR level. We represent modals by their lexical item (e.g., "∼2.mod=could" shows that the verb "∼2.is_a=write" has as modal *could*). Therefore, the valid questions that can be asked are those that use the same modal (we cannot ask a question with *can* or *should*, for example).

6.1.2 Nouns

Nouns are another well-studied lexical category in linguistic theories, both syntactically and semantically. Let us consider three examples:

(2) a. We enjoyed your wonderful concert on Saturday.

b. The interior of the concert hall is superb.

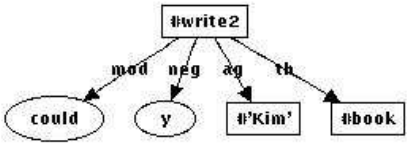
c. John's observation contradicted the hypothesis.

Kim could not have written the book.

TKR

```
~1.name='Kim'
~2.mod=could
~2.neg=y
~2.bse=have
~2.pf=y
~2.is_a=write
~2.ag=~1
~2.th=~3
~3.det=the
~3.is_a=book
```

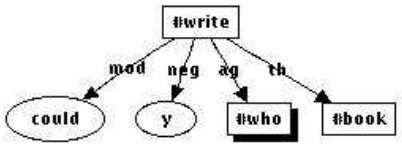
OKR



Questions

1. Who couldn't have written the book?
2. Who couldn't write the book?

OKR



Answer

```
[#'Kim']
```

OKR

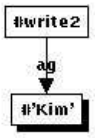


Figure 6.9: Representation and meaning for (1b)

While all three underlined lexical elements belong to the noun category, they assume a different syntactico-semantic representation. For example, *concert* in (2b) is a noun modifier, behaving similarly to adjectives, while the noun *observation* is a nominalization, behaving syntactically as a noun, but having the underlying semantics of a verb.

In Figure 6.10 we present the elementary semantic molecule templates needed for each of these phenomena.

The noun *concert* in constructions such as the one in (2a) has the elementary semantic molecule presented in Figure 6.10(a). The head of the semantic molecule (h) contains the following attributes: category attributes (cat), feature attributes (det, pers, nr, case,hum, count) and variable attributes (head). The category attribute specifies for lexical items their part of speech, in this case, noun.

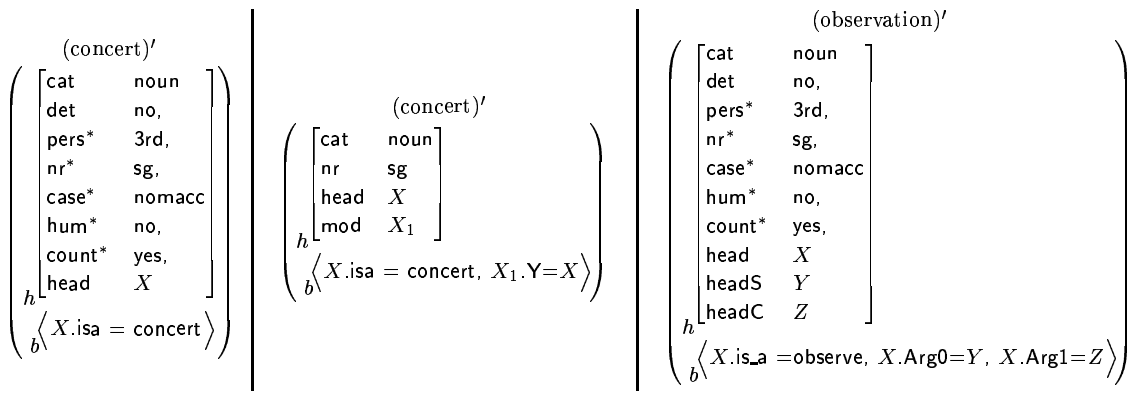


Figure 6.10: Elementary semantic molecules templates for three types of nouns: (a) basic nouns; (b) noun modifier; (c) nominalization

The feature attributes marked by $*$, as in the case of verbs, have as their values variables, which get instantiated for each lexical item. They are mainly agreement features. The feature attribute det no specifies that the bare common nouns are not determined. Proper nouns and pronouns have instead this feature attribute set to y, showing that a determiner cannot modify them. The variable attribute head specifies the index of the noun. The body of the semantic molecule (b), is the semantic representation (OntoSeR$^-$), which is a concept in the ontology.

The noun *concert* in the noun-noun compound construction in (2b) is a noun modifier, which has the semantic molecule presented in Figure 6.10(b). It behaves like an adjective, so that its semantic molecule is similar to that of adjectives (see Section 6.1.3 for adjective semantics). The head of the semantic molecule (h) contains only one feature attribute nr, which is set to sg, because noun modifiers cannot be plural. As variable attributes, we have both the usual head attribute, giving the index (X) of the lexical item (i.e., the modifier noun) and an additional attribute mod, which gives the index of the modified noun ($X1$). The body of the semantic molecule provides the semantics of the modifier noun. It is a concept in

an ontology (X.isa=concert), representing the value of a slot of another concept given by the modified noun, ($X1.Y{=}X$). In the example (2b) the modified concept is *hall* ($X1$) and the slot name is *purpose* (Y).

The noun *observation* is a nominalization, having the semantic molecule given in Figure 6.10(c). We noticed that the head of the semantic molecule contains all the attributes relevant to a noun, because nominalizations behaves syntactically as nouns. However, their underlying meaning is that of a verb. This fact is reflected both in the semantic representation, which is the representation of the corresponding verb (*observe*), and in the molecule head that has the same variable attributes as the ones of the corresponding verb (head, headS, headC). One aspect needs to be discussed here. The new type of semantic molecule for nominalization is required if we want to capture this phenomenon in the grammar. Another possibility would be to have the molecule of a basic noun, and to treat nominalization only at the interpretation level (TKR level). The important aspect is that our representation and our learning framework allow us to experiment with different alternatives.

Examples of TKR and OKR involving nouns will be given in the next sections when we introduce adjectives, coordinations, relative clauses as they will all present more complex constructions involving nouns.

6.1.3 Adjectives and Adverbs

In this dissertation, we rely mainly on the conventional wisdom that adjectives modify nouns and that they denote some properties of the concepts denoted by nouns.

In Figure 6.11(a) we present the elementary semantic molecule for adjectives, with an instance for the adjective *loud*. As can be seen, elementary semantic

$$(\text{loud})' = \left(\begin{array}{l} h\begin{bmatrix} \mathsf{cat} & \mathsf{adj} \\ \mathsf{head} & X \\ \mathsf{mod} & X_1 \end{bmatrix} \\ b\left\langle X.\mathsf{is\_a} = \mathsf{loud},\ X_1.\mathsf{Y}{=}X \right\rangle \end{array} \right) \qquad (\text{very})' = \left(\begin{array}{l} h\begin{bmatrix} \mathsf{cat} & \mathsf{adv} \\ \mathsf{mod} & X \end{bmatrix} \\ b\left\langle X.\mathsf{Y}{=}\mathsf{very} \right\rangle \end{array} \right)$$

Figure 6.11: Elementary semantic molecules templates for: (a) adjective; (b) adverb

molecules for noun modifiers and adjectives are almost identical, except that the molecule for adjectives does not need the feature attribute nr. In this way, we specify that the adjective is a concept in an ontology (X.isa=loud), which is the value of a slot (property) of a concept given by the modified noun ($X1$.Y=X). For example, *loud* is the value of the property *volume* (Y) for the concept *sound* ($X1$), in a noun phrase *loud sound*.

This representation is suitable for intersective adjectives (e.g., the concept *loud sound*, is still a *sound* that additionally has the property of being *loud*). There is another type of adjectives, non-interesective adjectives, for which this property does not hold (e.g., *former president* is not a *president*). In this dissertation, we are not concerned with the latter type of adjectives, but we can envision that they could be included with the same elementary semantic molecule, making the distinction between the two types at the interpretation level.

In Figure 6.11(b), we present the elementary semantic molecule for adverbs, with an instance for the adverb *very*. This elementary semantic molecule is a very shallow representation of adverbs. They are not concepts, but only values of slots of concepts. The mod feature is the index of the modified concept, we do not have the feature head for adverbs (they are headless, like determiners). The decision to adopt this representation for adverbs was a pragmatic one, since we do not encounter in our corpus complex constructions including adverbs (e.g., like adverb

I heard a very clear loud sound.

TKR

```
~1.is_a='I'
~2.tense=pt
~2.is_a=hear
~2.exp=~1
~2.perc=~3
~3.det=a
~4.degree=very
~4.is_a=clear
~3.clearness=~4
~5.is_a=loud
~3.volume=~5
~3.is_a=sound
```

OKR

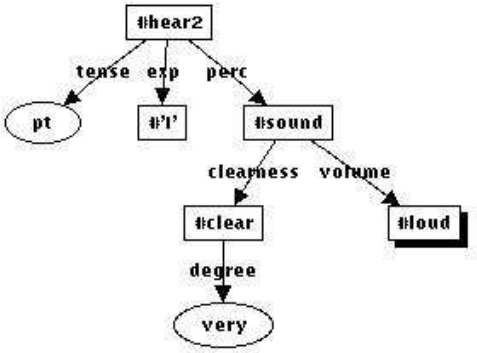


Figure 6.12: Representations for *I heard a very clear loud sound*

coordination that would require a representation for adverbs similar to the one for adjectives, with a head feature, and an enriched semantic representation). But this refinement can be performed, and will subsequently imply only the refinement of the representative examples involving adverbs.

What we want to mention is that the decision to refine elementary semantic molecules for different lexical items can be taken at any point. We provide an expressive representation and a grammar learning tool that makes the refinement task easy. Also, the underlying theoretical model provides a sound grammar revision.

An example of text and ontology level representations (TKR and OKR) of the sentence *I heard a very clear loud sound* is given in Figure 6.12. We treat adverbs in the same way as we treat determiners at the TKR level (i.e., they are "headless"). The difference is that we do not ignore them at the OKR level as we do with determiners.[4]

[4]The decision to ignore determiners is taken since the interpretation in this dissertation is focused on terminological knowledge.

$$(\text{of})' = \left(\begin{array}{l} h\left[\begin{array}{ll} \mathsf{cat} & \mathsf{prep} \\ \mathsf{head} & X \\ \mathsf{link} & Xl \\ \mathsf{comp} & Xc \end{array}\right] \\ b\left\langle X.\mathsf{is\_a} = \mathsf{of},\ Xl.X = Xc \right\rangle \end{array} \right) \qquad (\text{'s})' = \left(\begin{array}{l} h\left[\begin{array}{ll} \mathsf{cat} & \mathsf{iprep} \\ \mathsf{case} & \mathsf{g} \\ \mathsf{head} & X \\ \mathsf{mod} & Y \end{array}\right] \\ b\left\langle Y.\mathsf{of} = X \right\rangle \end{array} \right)$$

Figure 6.13: Elementary semantic molecules templates for: (a) prepositions; (b) genitive marker

6.1.4 Prepositions

Recently, there has been a growing awareness of the difficulties posed by prepositions and the importance of providing adequate means of capturing them, for many different applications (Jensen and Nilsson, 2003; Saint-Dizier, 2005).

Let us consider the sentence below:

(3) a. The president of Monsanto will give a briefing for the Brazilian media at noon.

What can be seen from this example is that prepositions express relations between two noun phrases (*of*), or between a verb and its complement (*for*), or adjunct (*at*). In this dissertation, we provide a uniform representation for prepositions, which accounts for all these three cases. The elementary semantic molecule template for prepositions is given in Figure 6.13(a), with an instance for the preposition *of*.

The head of the semantic molecule contains three variable attributes: head, which is the index of the preposition, link, which is the index of the concept to which the preposition is connected (either a noun or a verb), and comp, which is the index of the preposition complement. From the semantic representation we can see that prepositions are semantic roles, being slots in frames.

TKR

```
~8.det=the
~8.is_a=president
poss.is_a=of
~8.poss=~9
~9.name='Monsanto'
~10.mod=will
~10.is_a=give
~10.ag=~8
~10.th=~11
~10.goal=~12
~11.det=a
~11.is_a=briefing
goal.is_a=for
~10.goal=~12
~12.det=the
~13.is_a=brazilian
~12.origin=~13
~12.is_a=media
time.is_a= at
~10.time=~14
~14.is_a=noon
```

OKR

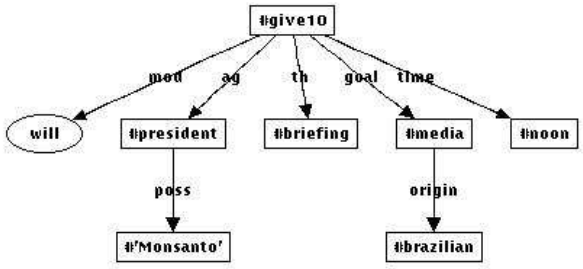


Figure 6.14: Representation of the example in (3)

From the semantic representation of the sentence in (3) given in Figure 6.14, the representation of prepositions becomes more clear. At the TKR level, we have the full representation of all three prepositions. The preposition *of* has in this example the representation ("poss.is_a=of", ~8.poss=~9), which says that *of* means "poss" (possession), and the concepts ~8 (*president*), ~9 (*Monsanto*) are in a semantic relation of possession. At the OKR representation, we have only the semantic role introduced by the preposition, and here we can clearly see that the preposition connects two concepts (i.e., two vertices in the directed graph), its semantics being the semantic role (i.e., the directed edge between the concepts).

In the grammar learned in this dissertation, we treat the genitive marker as the inverse of the preposition *of* from a semantic point of view. For example, *Monsanto's president*, is equivalent to *president of Monsanto*. The semantic molecule for *'s* is given in Figure 6.13. We have a feature attribute case which is set to gen-

$$(\text{and})'= \left(\begin{array}{l} \left[\begin{array}{ll} \mathsf{cat} & \mathsf{coord} \\ \mathsf{head} & X \\ \mathsf{linkl} & Y \\ \mathsf{linkr} & Z \end{array} \right] \\ h \\ b\left\langle X{=}Y \text{ and } Z \right\rangle \end{array} \right)$$

Figure 6.15: Elementary semantic molecule templates for coordination

itive. We have also two variable attributes head and mod. The head refers to the noun in the genitive (e.g., *Monsanto*) and mod refers to the modified noun (*president*). The intuition is that the construction *Monsanto's* behaves syntactically like an adjective that modifies a noun. Semantically, *'s* is the inverse of the preposition *of* (Y.of=X). Only at the interpretation level we will have the exact semantic role given by the preposition *of* in that particular context. This is because the semantic role of the genitive marker is not known a-priori.

6.1.5 Coordination

What we mean by coordination in this dissertation is a function that combines several sentence elements of the same or similar category into a single larger element. In our ontology-based approach, coordination can be applied to properties (encoded lexically by adjectives), relations (encoded either by prepositions or verbs), and entities (encoded by nouns). An example is given bellow:

(4) a. toxic **and** persistent contaminants on **and** in tomato **and** cucumber.

The lexical item that links the units of a coordinate construction is called *coordinator*. *And* is by far the most frequently occurring coordinator. However, coordinate constructions can also present various other types of linkers, such as *or*,

TKR

```
~1.is_a=toxic
~2.kind_of=~1
~3=~1 and ~4
~4.is_a=persistent
~2.permanence=~4
~2.is_a=contaminants
loc_surf.is_a=on
~2.loc_surf=~5
~6= loc_surf and loc_int
loc_int.is_a=in
~2.loc_int=~5
~7.is_a=tomato
~5=~7 and ~8
~8.is_a=cucumber
```

OKR

```
#contaminants.kind_of = #toxic
#contaminants.permanence = #persistent
#contaminants.loc_surf = [#tomato,and,#cucumber]
#contaminants.loc_int = [#tomato,and,#cucumber]
```

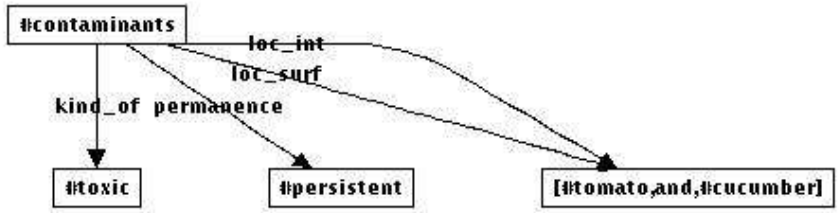


Figure 6.16: Semantic representation of coordination

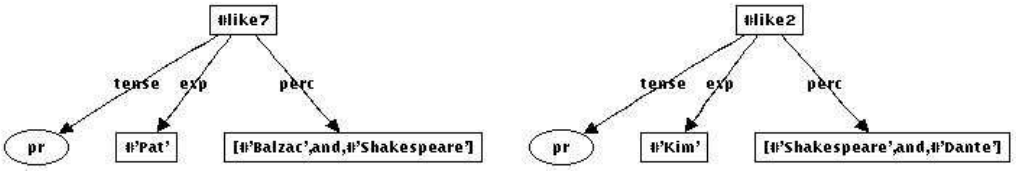
and *but*. *And*-coordination is also called conjunctive coordination (or conjunction), *or*-coordination is also called disjunctive coordination (or disjunction), while *but*-coordination is called adversative coordination.

In Figure 6.15 we give the elementary semantic molecule template for all these three coordinators, with an instance for *and*. Unlike the approach taken by (Copestake et al., 2001) we do consider coordinators as functions, not as binary relations. From the formal definition of OntoSeR given in Figure 6.1, we notice that coordination introduces a different definition for atomic predicate (AP), i.e., $\langle AP \rangle \stackrel{\text{def}}{=} \langle ConceptID \rangle = \langle ConceptID \rangle \;\; coord \;\; \langle ConceptID \rangle$.

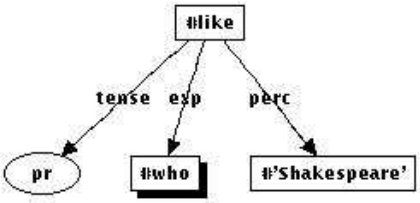
The body of the elementary semantic molecule template for *and* is given in Figure 6.15 (X=Y and Z), where X is the value of the head variable attribute, Y is

Kim likes Shakespeare and Dante.
Pat likes Balzac and Shakespeare.
OKR

```
#like2.tense=pr
#like2.exp= #'Kim'
#like2.perc=[#'Shakespeare',and,#'Dante']
#like7.tense=pr
#like7.exp= #'Pat'
#like7.perc=[#'Balzac',and,#'Shakespeare']
```



Question: Who likes Shakespeare?
OKR



Answer: #'Kim' , #'Pat'
OKR

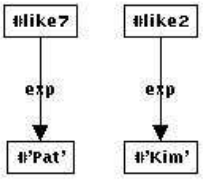


Figure 6.17: Multiple answers to questions

the index of the left coordinant (linkl), and Z is the index of the right coordinant (linkr). From this representation one can see that we treat coordinators as heads (Johannessen, 1997).

In Figure 6.16 we can see the TKR and OKR representation of the utterance given in (4), involving coordination between properties (adjectives *toxic*, *persistent*), relations (prepositions *on*, *in*) and entities (nouns *tomato*, *cucumber*). At the TKR level we have all the necessary information to perform the interpretation of coordination. At the OKR level, we keep the coordination of concepts as a logical proposition for further possible interpretation (e.g., [# tomato,and,#cucumber] in

Figure 6.16). In this dissertation, we interpret the conjunctive coordination as a set of concepts, while ignoring the coordinator. While we do represent the same the coordinators *but* and *or*, we leave for future research their specific interpretation.

Even for the conjunctive coordination, the set union is a very simplified way of interpretation. Let us take the following examples:

(5) a. young and clever man

b. *young and old man

c. young and old men

The first example enters under the most common case of *and* coordination as set union. If we look at the example (b) we have a contradiction: a man cannot be both young and old (we cannot have antonymic values of the same property assigned to the same concept). However, in (c) the coordination among *young* and *old* attached to the plural noun *men* implies the creation of two concepts: *young men* and *old men*. In this dissertation, we only treat the first two cases, but the interpretation can be extended to include the third case.

However, even if we do have a limited interpretation of coordination, this is sufficient to provide correct answers to questions involving coordinations, such as the case presented in Figure 6.17. Given the two utterances *Kim likes Shakespeare and Dante* and *Pat likes Balzac and Shakespeare*, when we ask the question *Who likes Shakespeare*, we get as answers both #Pat and #Kim.

In the grammar learned in this dissertation, we do not handle the coordination of different categories (*beautifully and with great empathy*), complex coordinators (e.g., *both ... and*), or ellipses. The same-category coordination is not

too restrictive, because we can introduce categories based on functional/semantic criteria and not on syntactic ones (for the example above we can have a category for manner adjuncts). We suggest that treatment of ellipses should be done at the robust parser level and not at the grammar level.

6.2 Raising and Control

In the previous section, we have described the semantics of lexical items, and shown the usefulness of our semantic molecule representation to model phenomena such as noun compounds, coordination, and nominalization. In this section, we present sentence-level constructions that express complex meanings, in which one situation functions as the semantic argument of another. Two such constructions are raising and control constructions.

6.2.1 Linguistic Phenomenon and Syntactic Analysis

Let us consider the sentences in (6), which have an infinitival complement:

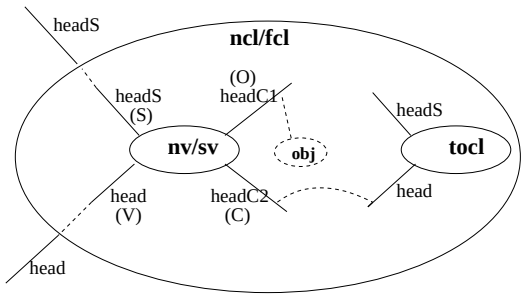
(6) a. Pat continues to avoid conflict.

b. Pat tries to avoid conflict.

Superficially these two constructions look the same. The embedded clause is missing the subject, and the subject (or the object as in (7)) of the matrix clause (i.e., the "controller") is interpreted as the subject of embedded clause.

(7) a. We expect the doctor to examine us.

b. I persuade the doctor to examine us.



fcl - finite clause
ncl - nonfinite clause
tocl - to-infinitive clause (subsumed by **ncl**)
sv - finite verb
nv - nonfinite verb
obj - object

(a)

vtype	ncl/fcl		nv/sv				tocl	
	head	headS	head	headS	headC1	headC2	head	headS
rsbj	V	S	V	S	O	C	C	S
robj	V	S	V	S	O	C	C	O
csbj	V	S	V	S	O	C	C	S
cobj	V	S	V	S	O	C	C	O

(b)

Figure 6.18: Semantic molecules linking in raising and control constructions

The essential difference is semantic. In constructions such as (6a) and (7a) the controller is not one of the semantic arguments of the matrix verb. These are called raising constructions (raising to subject, and raising to object, respectively). In constructions such as (6b) and (7b) the controller is one of the semantic arguments of the matrix verb. The constructions are called control (subject control, and object control, respectively). Both in raising and control constructions, the controller is one of the arguments of the embedded verb.

Like in HPSG (Pollard and Sag, 1994; Sag and Wasow, 1999), there is no structural difference between raising constructions and control constructions in our approach. For example, in both subject raising and subject control constructions (6), the controller is realized by the syntactic subject of both the matrix verb and the embedded verb. What constitutes a major difference in our approach is the

fact that we do not have different elementary semantic molecules (i.e., structures of lexical items) for raising and control verbs. They have elementary semantic molecules like any other verb, the difference being set by the value of the attribute vtype: rsbj, robj,csbj,cobj, and by their thematic roles (e.g., subject raising verbs have a no_role role, encoding the fact that these verbs do not have an agent-like semantic argument).

In our framework, the specific constructions for subject/object raising and subject/object control are encoded by grammar rules that are learned from examples.

Figure 6.18 presents schematically the type of semantic molecules that are involved in raising and control constructions and how linking (i.e., composition) is performed.[5] The principal semantic molecules are: sv which stands for the finite verb together with its subject (including agreement), tocl the infinitival complement (i.e., *to*-infinitive clause), and fcl which is the finite clause. To give an example, the sentence in (7b) is composed from the following molecules:

$$\overbrace{\underbrace{\text{I persuaded}}_{sv}\,\underbrace{\text{the doctor}}_{obj}\,\underbrace{\text{to examine us}}_{tocl}}^{fcl}$$

In order to obtain recursive constructions, we can have nonfinite clauses, ncl, instead of finite clauses, and nonfinite verbs nv, instead of finite verbs. Recursive constructions will allow us to process sentences like *We tried to convince him to try to quit smoking*, or *I saw Mary trying to convince John to sell the house.*

[5]This figure exemplifies constructions where the matrix verb is ditransitive. For transitive constructions, the verb has only one complement, and thus we would have headC instead of headC1 and headC2.

Figure 6.18 presents only the elements of the semantic molecule heads that are involved in the composition process (i.e., the variable attributes). The nv/sv molecule has four variable attributes: head, the head of the molecule, which is the index of the raising/control verb; headS, the index of the subject of the raising/control verb; headC1, the index of its direct complement, and headC2, the index of the indirect complement. The tocl molecule has only two variable attributes: head and headS, because only these are relevant in the linking, the complements of the embedded clause (*to*-infinitive clause) being already consumed (linked). From the table in Figure 6.18 it can be seen that the head and headS of the finite/nonfinite resulting clause is always the same as the head and headS of the nv/sv molecule. Also the headC2 complement is always the same with the head of the tocl molecule. The difference in subject raising/control versus object raising/control is the way in which the subject(headS) of the to-infinitive clause is linked. In subject raising/control it is linked to the subject headS of the nv/sv molecule, while in object raising/control it is linked to the complement headC1 of the nv/sv molecule. The linking (i.e., composition) is performed through the semantic composition constraint Φ_{comp}, which is encoded as path equations learned together with the grammar rules (see Appendix A.4). The advantage of our semantic molecule representation can be seen in these complex constructions, since we keep in the head of the semantic molecules all the mandatory information required for further composition.

As can be seen from Figure 6.18 raising and control are similar structurally and compositionally. What, then, is the difference between raising predicates, such as *continue* or *expect*, and control predicates, such as *try* or *persuade*? The essential difference is semantic: the controller arguments of control predicates are

assigned a semantic role by the control verb, while the controller arguments of raising predicates are not assigned such a role by the raising verb. These semantic differences will become clear in the next sections, with the presentation of the semantic representation of these phenomena.

6.2.2 Semantic Analysis

6.2.2.1 Subject Raising and Subject Control

As we mentioned in the Section 6.2.1, syntactically we treat raising and control constructions the same. That is, Pat is the subject of the matrix verb in both of the constructions bellow.

(8) a. Pat continues to avoid conflict.

b. Pat tries to avoid conflict.

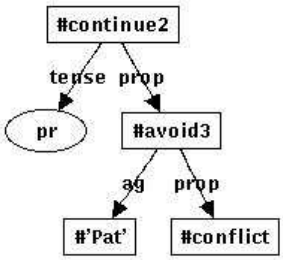
However, semantically there is a difference: Pat is not the "agent" (ag) of the raising verb *continue*, but is the agent of the control verb *try*. In our framework this is implemented by having the raising verb specifying a no_role role. Even if at the text-level representation (TKR), this information is represented (∼2.no_role=∼1), at the OKR level this information is filtered and we obtain the representation as given in Figure 6.19. For the control verb *try*, we have an agent role ag, which is kept as usual at the OKR representation. The fact that these two constructions are semantically different is captured also by the fact that for raising verbs, we can ask either *What does Pat continue?* and *What continues?*, while for control constructions we cannot ask the equivalent to the last question *What tries?*, since *Pat* is the agent of the matrix verb.

Pat continues to avoid conflict.

TKR

```
~1.name='Pat'
~2.tense=pr
~2.is_a=continue
~2.no_role=~1
~2.prop=~3
~3.vft=to
~3.is_a=avoid
~3.ag=~1
~3.prop=~4
~4.is_a=conflict
```

OKR



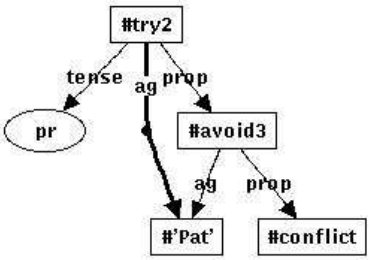
(a1)

Pat tries to avoid conflict.

TKR

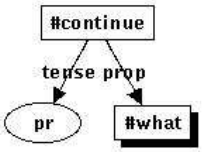
```
~1.name='Pat'
~2.tense=pr
~2.is_a= try
~2.ag=~1
~2.prop=~3
~3.vft=to
~3.is_a=avoid
~3.ag=~1
~3.prop=~4
~4.is_a=conflict
```

OKR

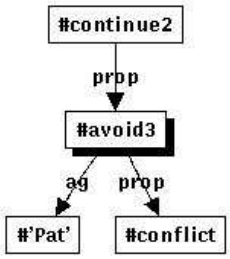


(b1)

Question(s):
What does Pat continue?
What continues?

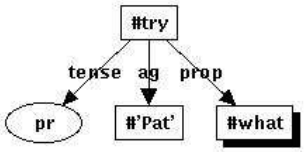


Answer:[#avoid3]

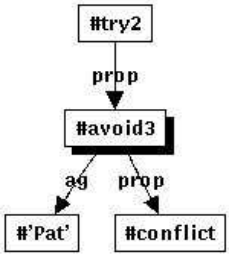


(a2)

Question:
What does Pat try?



Answer:[#avoid3]



(b2)

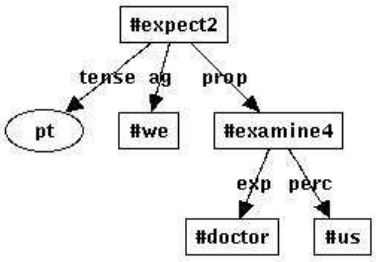
Figure 6.19: Semantics of subject raising and subject control constructions

We expected the doctor to examine us.

TKR

```
~1.is_a=we
~2.tense=pt
~2.is_a=expect
~2.ag=~1
~2.no_role=~3
~2.prop=~4
~3.det=the
~3.is_a=doctor
~4.vft=to
~4.is_a=examine
~4.exp=~3
~4.perc=~5
~5.is_a=us
```

OKR



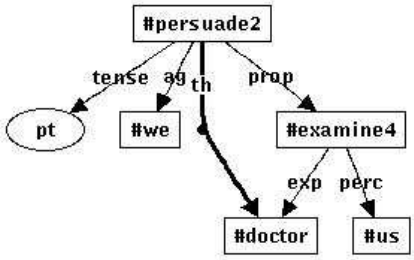
(a)

We persuaded the doctor to examine us.

TKR

```
~1.is_a=we
~2.tense=pt
~2.is_a=persuade
~2.ag=~1
~2.th=~3
~2.prop=~4
~3.det=the
~3.is_a=doctor
~4.vft=to
~4.is_a=examine
~4.exp=~3
~4.perc=~5
~5.is_a=us
```

OKR



(b)

Figure 6.20: Semantics of object raising and object control constructions

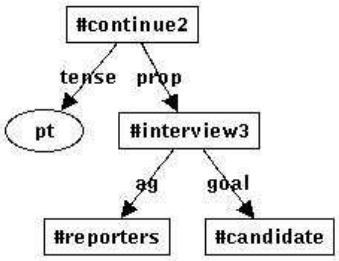
6.2.2.2 Object Raising and Object Control

For object raising/control construction the difference is that the "object" of the matrix verb is the theme (th) for control verbs, but is not the theme for the raising verbs. Thus, the raising verb has a no_role semantic role, similar to the no_role role in the subject raising constructions. The semantic difference between object raising and object control can be clearly perceived from the OKR representation of the two constructions given in (9), given in Figure 6.20.

(9) a. We expect the doctor to examine us.

Reporters continued to interview the candidate

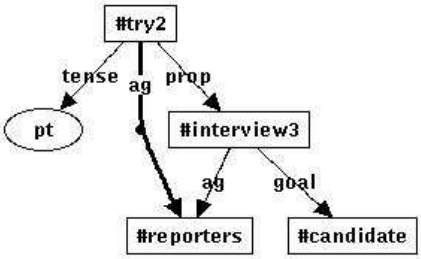
OKR



(a1)

Reporters tried to interview the candidate.

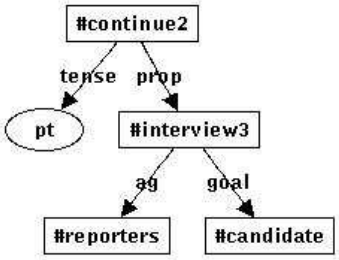
OKR



(b1)

The candidate continued to be interviewed by reporters.

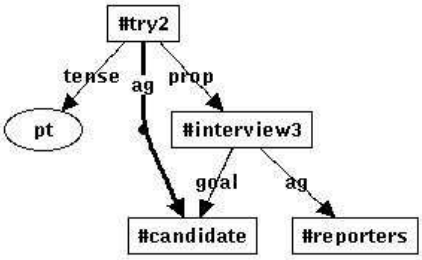
OKR



(a2)

The candidate tried to be interviewed by reporters.

OKR



(b2)

Figure 6.21: Passivizing the complement of the matrix verb: *continue* (a1,a2); *try* (b1,b2)

b. We persuade the doctor to examine us.

6.2.2.3 Passivizing the Complement of the Matrix Verb

In the previous sections, we analyzed the semantic difference between raising and control construction. Another semantic test showing this difference is the passivization of the complement of the matrix verb. For raising constructions, passivizing the complement does not change the truth conditions, while for control construction it does. That is, the sentences in (10) are semantically the same, namely they

are paraphrases, while the sentences in (11) are not. Our system can predict this phenomenon. From Figure 6.21(a1) and 6.21(a2) it is clear that the OKR of (10a) is the same as the OKR of (10b). For the control constructions (11) this is not the case, given that in the OKR of (11a) the agent of *try* is *reporters* (Figure 6.21(b1)), while in the OKR of (11b) the agent of *try* is *candidates* (Figure 6.21(b2)).

(10) a. Reporters continued to interview the candidate.

b. The candidate continued to be interviewed by reporters.

(11) a. Reporters tried to interview the candidate.

b. The candidate tried to be interviewed by reporters.

Similar analyses can be done for the sentences (12) and (13), taken from (Sag, Wasow, and Bender, 2003). Moreover, from these examples, it is clear that the controller must be the subject of the raising verb since agreement needs to be performed (*skeptics continue* vs. *your hypothesis continues*).

(12) a. Skeptics continue to question your hypothesis.

b. Your hypothesis continues to be questioned by skeptics.

(13) a. The police tries to arrest disruptive demonstrators.

b. Disruptive demonstrators try to be arrested by the police.

6.3 Relative Clauses and Wh-Questions

An interesting natural language phenomenon is the existence of long-distance dependencies between phrases that are not adjacent in the surface sequence of words. Long-distance dependencies can arise in at least two ways (Philips, Kazanina, and

Abada, 2005), as shown (14):

(14) a. Monsanto's president [who seems to try to get royalties from the farmers [who grew genetically modified soya illegally]] will be giving a briefing for the Brazilian media tomorrow at noon .

b. What does the president seem to try to get _ from the farmers?

One situation, given in (14a), involves subject-verb agreement relations, where the elements are structurally close to one another and they are in their canonical position, but are separated by additional material (e.g., relative clauses).

Long-distance dependencies also arise when a phrase is moved from its canonical position to a potentially unbounded distance from that position. This situation appears both in relative clauses and in wh-questions. For example, the canonical position of a direct object is immediately after the verb that assigns its thematic role (e.g., *royalties* in (14a)). But in a wh-question, this direct object NP, which becomes a wh-word, is moved to the front of the clause as in (14)b. Following standard linguistic terminology we call the *wh*-word the *filler* and the canonical position of the fronted NP, marked by underlining in (14b), the *gap*. The combination of the two is known as *filler-gap dependency* (Fodor, 1978). In Section 6.3.3 we show that in this dissertation, filler-gap dependencies are a processing effect, that is, they are resolved during parsing by our robust parser.

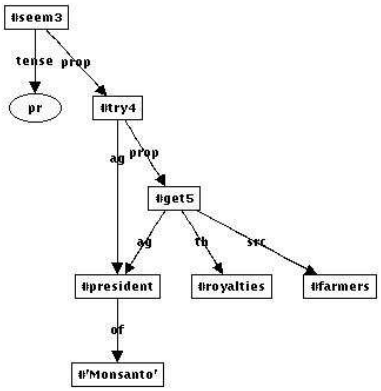
Relative clauses and wh-questions are both wh-clauses. Without further context, the clause *who visited Kim* can be either a relative clause or a wh-question. In this dissertation, we introduce a feature attribute int(interrogative) at the level of the semantic molecule head, which is underspecified by default. When this clause is

Monsanto's president who seems to try to get royalties from the farmers.

TKR

```
^1.name = 'Monsanto'
^2.of =^1
^2.is_a = president
^2.is_a = who
^3.tense = pr
^3.is_a = seem
^3.no_role =^2
^3.prop =^4
^4.vft = to
^4.is_a = try
^4.ag =^2
^4.prop =^5
^5.vft = to
^5.is_a = get
^5.ag =^2
^5.th =^6
^5.src =^7
^6.is_a = royalties
src.is_a = from
^7.det = the
^7.is_a = farmers
```

OKR



The toy which Pat handed to the baby.

TKR

```
^1.det = the
^1.is_a = toy
^1.is_a = which
^2.name = 'Pat'
^3.tense = pt
^3.is_a = hand
^3.ag =^2
^3.th =^1
^3.goal =^4
goal.is_a = to
^4.det = the
^4.is_a = baby
```

OKR

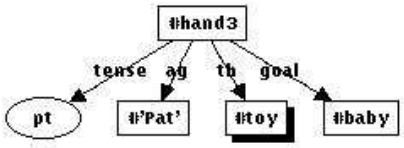


Figure 6.22: Representation of wh-relative clauses

attached to a noun, the value of int feature is no, while when we have a wh-question it is y. An example illustrative for this phenomenon is given in Appendix A.3.1 for the utterance *who can not have been going*. In the next two subsections, we will give several examples of how relative clauses and wh-questions are represented in our framework.

6.3.1 Relative Clauses

In this section we focus mainly on the semantic representation of relative clauses, both at the text-level and ontology-level.

Regarded syntactically, relative clauses are postmodifiers of nouns. Thus, the head of the relative clause molecules has a variable attribute mod, which will be linked to the modified noun, similar to the mod attribute used for adjectives. But the *wh*-words do not have this feature, since the wh-clause becomes a relative clause, only when it is attached to a noun, as we mentioned before. We handle the agreement between the head noun and the verb in the relative clause. The proper treatment of agreement helps in disambiguation as can be seen in *the two endocrine glands [located above the kidney] [that secrete hormones and epinephrine].* The second relative clause is connected to the noun *glands* and not *kidney*, because the verb is in the plural.

In (15) we present the types of relative clauses that are treated in this dissertation: subject wh-relatives (15a), nonsubject wh-relatives (15b), *that*-relatives (15c), reduced relative clauses (15c,d). *That*-relatives are treated as wh-relatives in our approach, given that we consider *that* as a *wh*-pronoun as in (Sag, 1997).

(15) a. Monsanto's president [who seems to try to get royalties from the farmers]

b. the toy [which Pat handed to the baby]

c. an acute viral hepatitis [caused by a virus [that tends to persist in the blood serum]]

d. an inflammatory disease [involving the sebaceous glands of the skin]

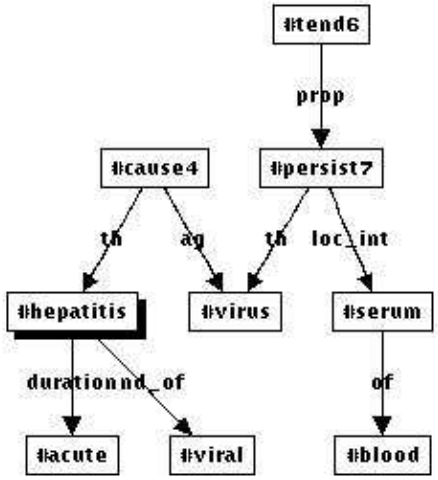
Figure 6.22 shows the text-level and ontology-level representations of the wh-relative clauses (15a, b). From the text-level representation (TKR) we can see that the *wh*-word has the same concept ID with the noun concept it modifies, and both

an acute viral hepatitis caused by a virus that tends to persist in the blood serum

TKR

```
~1.det= an
~2.is_a= acute
~1.duration=~2
~3.is_a= viral
~1.kind_of=~3
~1.is_a= hepatitis
~4.vft= ed
~4.voice= pas
~4.is_a= cause
~4.ag=~5
~4.th=~1
ag.is_a= by
~4.ag=~5
~5.det= a
~5.is_a= virus
~5.is_a= that
~6.tense= pr
~6.is_a= tend
~6.no_role=~5
~6.prop=~7
~7.vft= to
~7.is_a= persist
~7.th=~5
loc_int.is_a= in
~7.loc_int=~8
~8.det= the
~9.is_a= blood
~8.of=~9
~8.is_a= serum
```

OKR



an inflammatory disease involving the sebaceous glands of the skin

TKR

```
~1.det= an
~2.is_a= inflammatory
~1.kind_of=~2
~1.is_a= disease
~3.vft= ing
~3.is_a= involve
~3.exp=~1
~3.perc=~4
~4.det= the
~5.is_a= sebaceous
~4.kind_of=~5
~4.is_a= glands
of.is_a= of
~4.of=~6
~6.det= the
~6.is_a= skin
```

OKR

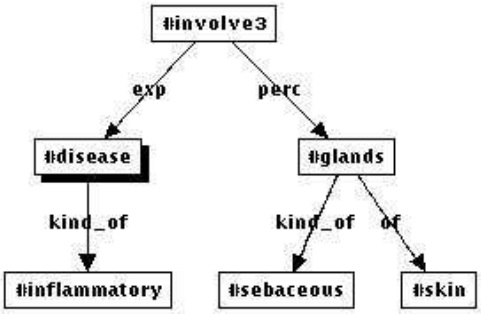


Figure 6.23: Representation of reduced relative clauses

fill the appropriate verb argument position. At the ontology-level representation (OKR), the *wh*-word is not represented, and the concept given by the noun fills the appropriate verb argument position.

Figure 6.23 presents the text-level and ontology-level representations of the reduced relative clauses (15c, d). Reduced relative clauses do not have an overt *wh*-word, and they are similar to the non-finite participial clauses (*-ed* and *-ing* clauses), but they modify a noun.

6.3.2 Wh-Questions

In order to illustrate several types of wh-questions let us consider the sentence given in (14a). This is a complex example that contains embedded relative clauses and raising and control constructions. The ontology-level representation (OKR) is given in Figure 6.24.

Let us consider the following questions, which exemplify different gapping positions:

(16) a. Who grew genetically modified soya?

b. What did the farmers grow illegally?

c. For whom will Monsanto's president be giving a briefing?

We can ask questions for relative clauses (16a,b). An instance of a question involving an indirect object as answer is illustrated in (16c). The ontology-level representation of both questions and answers is given in Figure 6.25. The answer is done by graph matching, and this can be easily seen when trying to map the OKR of the question onto the OKR of the sentence. The *wh*-word in question matches

Monsanto's president who seems to try to get royalties from the farmers who grew genetically modified soya illegally will be giving a briefing for the Brazilian media tomorrow at noon .

OKR

```
#president.of=#'Monsanto'
#seem3.tense=pr
#seem3.prop=#try4
#try4.ag=#president
#try4.prop=#get5
#get5.ag=#president
#get5.th=#royalties
#get5.src=#farmers
```

```
#grow8.tense=pt
#grow8.ag=#farmers
#grow8.th=#soya
#grow8.manner=illegally
#soya.quality=#modified
#modified.viewpoint=genetically
```

```
#give11.mod=will
#give11.ag=#president
#give11.th=#briefing
#give11.goal=#media
#give11.temporal=tomorrow
#give11.time=#noon
#media.origin=#brazilian
```

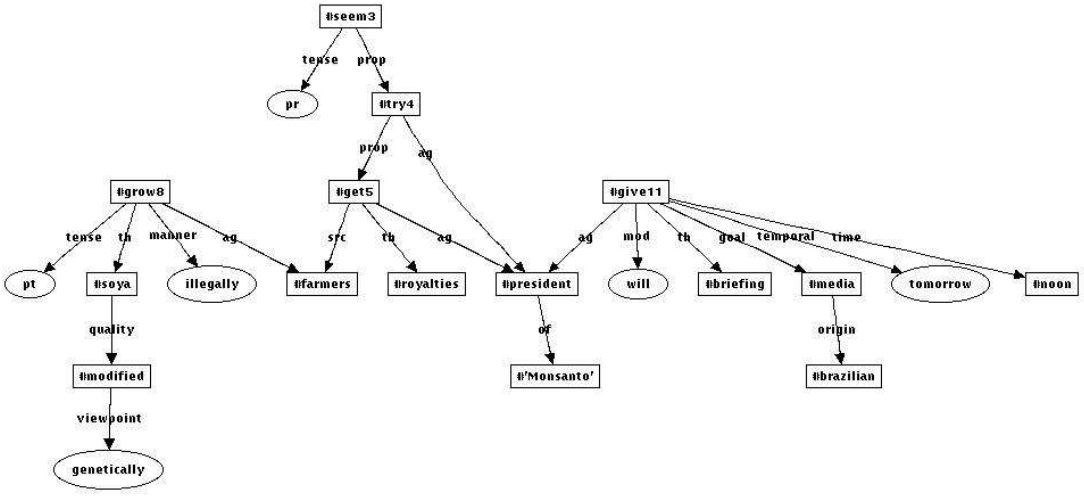
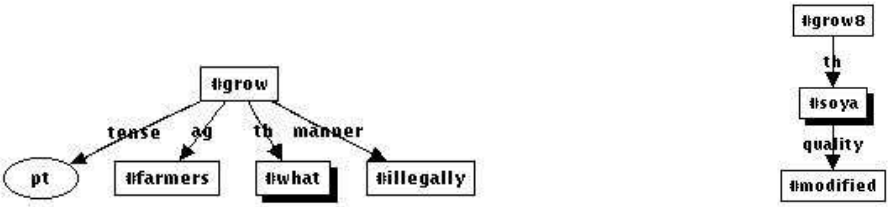


Figure 6.24: OKR representation for the sentence in (14a)

Q: Who grew genetically modified soya?
A: #farmers

Q: What did the farmers grow illegally?
A: #soya



Q: For whom will Monsanto's president be giving a briefing?
A: #media

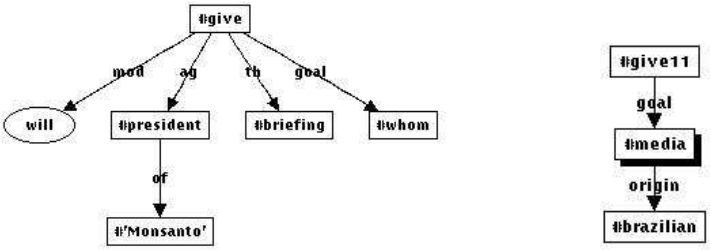


Figure 6.25: Questions and answers for the sentence in (14a) and Figure 6.24

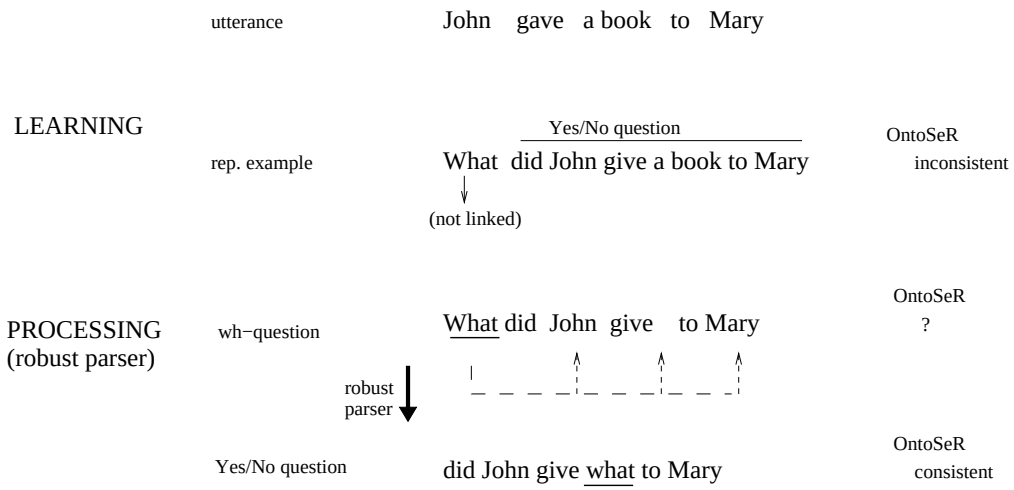


Figure 6.26: Learning and processing of filler-gap dependencies

the concept in the sentence. The answer consists of the concept node together with all its edges.

6.3.3 Filler-Gap Dependencies

In this section, we present our mechanism for *filler-gap dependencies*. These constructions are often instances of long-distance dependencies, like the wh-question in (17):

(17) a. What does the president seem to try to get _ from the farmers?

In this dissertation, we treat filler-gap dependencies as processing effects, that is, they are resolved during parsing by our robust parser. The key element which triggers this view is learning. Since we are concerned with learning the grammar and the grammar rules, it seems impossible to think of learning rules that would capture all the situations where gapping occurs.

In Figure 6.26 we present an example for a wh-question that illustrates both the learning and the processing phase for filler-gap dependencies. Let us consider the utterance *John gave a book to Mary.* In order to learn grammar rules for filler-gap constructions (in this case wh-questions), we use resumptive pronouns, or resumptive NPs in lieu of the gap. For example, the representative example from which the rule is learned is *What did John give a book to Mary*, where *a book* is a resumptive NP. The *wh*-word is not linked, and the construction is in fact a *wh*-word followed by a full Yes/No question. During learning we do not have wh-movement, and the use of resumptive pronouns/NPs allows us to learn general rules without considering every possible position of a gap. At this stage, OntoSeR is inconsistent, since we have an unlinked variable (the one corresponding to the *wh*-word). While we are not claiming the adequacy of our framework for modeling child language acquisition, it is worth noticing that the use of resumptives has been noticed in several studies of child acquisition of relative clauses (Diessel and Tomasello, 2005). Labelle (1990) argues that the use of resumptives is incompatible with wh-movement.

During parsing, the robust parser tries to fill subsequently all the verb argument positions (including the ones of the verbs from embedded clauses), until the OntoSeR becomes consistent, that is, until no variable remains unlinked. We show this parsing process in Figure 6.26. The results will be a consistent Yes/No type question, where the *wh*-word fills the right verb argument (e.g., *did John give what to Mary.* In Appendix A.5 we show the treatment of long distance dependencies through an example, for which we give the OntoSeR^{-} (representation returned by the robust parser before applying the consistency checks performed by Φ_{onto}),

Q: What does the president seem to try to get from the farmers?
A: #royalties

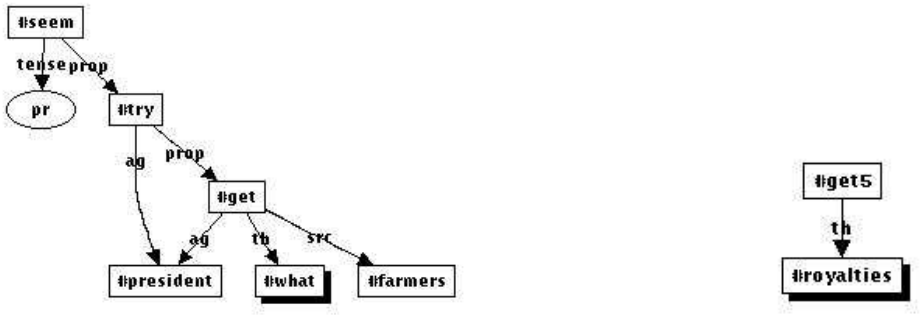


Figure 6.27: OKR for an example of long-distance dependency

OntoSeR$^+$, the TKR and the OKR representations.

It can be noticed that the "filler-gap" dependency can be very far when we also have raising and control constructions as in (17a). An OKR representation is given in Figure 6.27 together with the answer concept (Appendix A.5).

6.4 Ambiguity

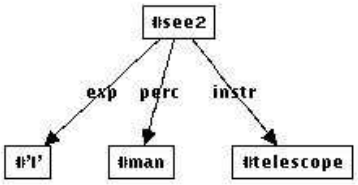
We saw throughout this dissertation that we do not have syntagma ambiguities (syntagmas are pairs of utterances and their semantic molecules). This is essential for the grammar learning process, where the representative examples and the representative sublanguage are actually syntagmas. During language processing, more exactly during parsing, we have unannotated utterances, and thus ambiguity can occur. We can have many TKRs/OKRs corresponding to the same utterance. In this case, our robust parser provides all alternatives. Let us consider the classical example:

(18) a. I saw the man with the telescope.

TKR

```
~1.is_a= 'I'
~2.tense= pt
~2.is_a= see
~2.exp=~1
~2.perc=~3
~3.det= the
~3.is_a= man
instr.is_a= with
~2.instr=~4
~4.det= the
~4.is_a= telescope
```

OKR



TKR

```
~5.is_a= 'I'
~6.tense= pt
~6.is_a= see
~6.exp=~5
~6.perc=~7
~7.det= the
~7.is_a= man
instr.is_a= with
~7.instr=~8
~8.det= the
~8.is_a= telescope
```

OKR

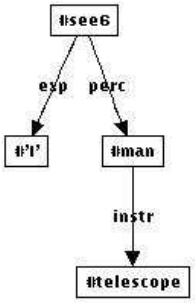


Figure 6.28: Two TKRs and OKRs for *I saw the man with the telescope*

From Figure 6.28 we can see that this utterance has two text level representations (TKRs) and two corresponding ontology level representations (OKRs). This is possible since we do have two grammar rules from which this utterance can be ground-derived, and the compositional constraints and the ontology constraints satisfies both these alternatives. The ambiguity can be eliminated in this case only if we have a discourse context. In this case, we would have two TKRs but only one OKR representation, since the interpretation from the TKR to the OKR might consider discourse context.

We have a similar case for the sentence in Figure 6.24, where *illegally* can be attached either to the verb *grow*, or to the verb *get* (again, the reason is that

the utterance can be derived from two grammar rules (i.e., to-infinitive clause + adjunct and relative clause + adjunct) and the constraints satisfy both of these alternatives).[6] We present this ambiguity example in Appendix B.

However there are cases where ambiguities can be eliminated by the use of our grammar constraints:

(19) a. the two endocrine glands [located above the kidney] [that secrete hormones and epinephrine]

b. I saw the man with the blue shirt.

In the first example the second relative clause can be attached to the noun *kidney* or the noun *glands*. Since we have agreement between the head noun and the verb in the relative clause, we have that the relative clause is attached to the noun *glands* (plural). This is achieved through our compositional constraint Φ_{comp}. In the second example, the ambiguity can be eliminated through semantic interpretation given a strong semantic context that has hierarchies of concepts and roles, as well as selectional restrictions. This way, the Φ_{onto} constraint, based on this strong semantic context, allows only one interpretation: the prepositional phrase *with the blue shirt* is associated with the noun *man* and not with the verb *saw*. In the same way, polysemy can be handled. In this dissertation, we did not experiment with a strong context (i.e., a full-fledged ontology), but such resources, when available, can be incorporated in our framework, since our representation is an ontology query language, as this chapter has shown. In Chapter 8 we will show that this representation is useful for acquiring ontological knowledge from natural language.

[6] A *to*-infinitive clause allows an adjunct only at the innermost level.

Chapter 7

A General Framework for LWFG Learning

Part I of this dissertation has presented a major *theoretical result*:

> Lexicalized Well-Founded Grammars can always be learned from a set of semantically annotated representative examples and a representative sublanguage (see Theorems 2, 3, and 8).

As a consequence, the following statement holds true:

> If natural language can be covered by Lexicalized Well-Founded Grammars, natural language can be learned.

While a major finding in itself, this theoretical result triggers an intriguing question *Can natural language be covered by LWFGs?*, which in our framework translates to *Does the Grammar Approximation by Representative Sublanguage (GARS) model guarantee an approximation of natural language by LWFGs?*

A positive answer about the GARS model can be obtained only empirically through an extensive quantitative evaluation on large corpora. Although our learning is based on a small annotated treebank, for a broad coverage quantitative

evaluation we would need a large OntoSeR-annotated treebank. Unfortunately, such a resource is not currently available, in order to perform such a quantitative evaluation.

In this dissertation, we have learned an experimental LWFG. Since it is clear that we cannot provide a definite positive answer to the above question, we have performed experiments to show the quality of our learned LWFG, and of our semantic representation OntoSeR with respect to:

- Coverage of difficult and diverse linguistic phenomena (e.g., raising and control, noun compounds, nominalization, long-distance dependencies, complex noun phrases, coordination, complex verbal constructions with auxiliaries).
- Usefulness of our semantic representation for direct knowledge acquisition from text.

In Chapter 6 we have presented the first quantitative evaluation, while Chapter 8 we will present the second. In order to carry out these evaluations we needed: 1) a general framework for LWFG learning; and 2) a semantic interpreter. In this chapter we present the general framework for LWFG learning, while in Chapter 8 we will present our semantic interpreter.

The core of our grammar-learning approach is the fact that the process of writing grammar rules and constraints by hand is replaced by the construction of a small annotated treebank — utterances and their semantic molecules. This treebank consists of: 1) a set of representative examples, E_R (simplest examples used to generate the grammar rules) and 2) representative sublanguage E_σ (used for performance criteria in rule generalization). Our grammar-learning framework has the

practical advantage of allowing grammar revision through an iterative algorithm, as we have shown in Chapter 5. Thus, if the grammar needs to be refined, we only refine the representative examples/sublanguage, and not the grammar rules and the constraints, which would be a more difficult process. In this dissertation, we implemented a system that represents an experimental platform for all theoretical algorithms presented in Part I. This system consists of:

- an Inductive Logic Programming system that learns only from positive examples, uses background knowledge and has a dual mode of operation: learns both from ordered and unordered examples (iterative algorithm).
- a robust parser/generator that has been used in all our learning experiments to signal the over/under generalization and ambiguity.

In the remainder of this chapter we present the details of learning an experimental LWFG which we evaluate qualitatively in this dissertation (Chapter 6 and Chapter 8).

7.1 Learning an Experimental LWFG

Chapter 6 presented the linguistic phenomena that our learned experimental grammar covers, and has shown the degree of expressiveness of our semantic representation with regard to those phenomena. In this section, we describe what was needed for learning this grammar (the training data), the overall results we have obtained, and a discussion of some controlled experiments used to further demonstrate the practical venue of our theoretical work.

Lexical Categories	Elem. Sem. Mol.		Nonterm.	Attributes			$\|E_R\|$	$\|E_\sigma\|$
	Types	Templates		Feature	Variable	Constant		
13	25	46	31	24	12	1	151	448

Table 7.1: Statistics of the input data for learning

7.1.1 Training Data

Table 7.1 presents the statistics of the training data required for learning. Regarding the lexical items, we have a total number of 13 lexical categories (i.e., preterminals, or parts of speech), 46 elementary semantic molecule templates that represent 24 types. For example, as we mentioned in Section 6.1.1, the verbs have 5 types of elementary semantic molecules, which gives a total number of 22 different templates (e.g., the type vtnsSem (finite, tensed verb) has three different templates for intransitive/transitive/ditransitive). More details on the lexical item statistics and the elementary semantic molecule templates have been presented in Chapter 6, Section 6.1.

In order to learn our grammar, we need to provide the names of the new categories which, as we said in our previous chapters, give the names of the grammar nonterminals. As Table 7.1 shows, we have 31 categories/nonterminals, which are different from the lexical categories/preterminals. We also need to provide the attributes that appear in the semantic molecule heads of all the categories (including lexical categories). We have 24 feature attributes, 12 variable attributes and 1 constant attribute (cat, whose values are the categories). The last two columns give the size of our semantically annotated treebank that was required for learning. We have two sets of positive examples: 1) representative examples, used to generate the grammar rules (one example per rule), and 2) representative sublanguage used

Gr. No	NL fragment	$\|E_R\| = \|P_G\|$	$\|E_\sigma\|$	$\|N_{lhs}\|$	$\{N_{lhs}\}$
1	fin. aux (+sbj+agr)	14	81	6	{sbj, av0, av1, av2, av3, av4}
2	nonfin aux	5	12	2	{nav0, nav1}
3	nonfin verbs	35	88	5	{nv, nvo, snvo, obj, pobj}
4	fin verbs (+sbj+agr)	28	115	2	{sv, svo}
5	noun phrases	26	53	10	{sbj, obj, pobj, a, na, nc, n, d, p, pc}
6	fin + nonfin clause (raising/control)	16	41	4	{fcl, ncl, tocl, adjc}
7	relative clause	10	25	2	{relcl, n}
8	wh-questions	7	16	1	{whcl}
9	copula "to be"	4	6	2	{npred, fcl1}
10	reduced relative clause	6	11	2	{rrcl, n}
Total		151	448	31 + (5)	

Table 7.2: Statistics of our learned experimental grammar

for generalization. In order to learn our grammar, we annotated 151 representative examples E_R and 448 examples that constitute the representative sublanguage E_σ. In this experiment, since both E_R and E_σ are fully annotated, ontological validation is not needed — the annotation provides the disambiguation required for learning, i.e, syntagma unambiguity. Only a reduced lexicon is needed for training (e.g., only a few lexical items are given for every open word class, such as nouns (20), verbs (13, 6 of which are for raising and control verbs), adjectives (14), adverbs (9), proper nouns (4)). For testing we built a larger lexicon required for various experiments (see Section 8.4 for details).

7.1.2 Overall Results

The summary of the linguistic phenomena covered by the learned experimental grammar is given in Table 7.2. We have an incremental learning methodology due to a practical reason, which is the time of learning. Execution time depends on the

size of the representative sublanguage set, $|E_\sigma|$.[1] Thus, E_σ, and accordingly E_R, have been fragmented into 17 groups of treating the linguistic phenomena presented in Table 7.2.[2] In the figure we give only the 10 main groups (the finite verbs have been divided in 4 subgroups, while the non-finite verbs in 5 subgroups). Another reason for learning incrementally from smaller groups is that the overall size of E_σ is in this case smaller than if we would learn from the whole E_R at once.[3] This practical importance of incremental learning emphasizes once more the theoretical assets of our learning framework: sound grammar revision and merging methods. In the remainder of this section, we present the learned experimental grammar, detailing each group of linguistic phenomena mentioned in Table 7.2.

1. **finite auxiliary verbs.** To learn auxiliary constructions, including modals, negation, tense, aspect, periphrastic *do*, we used 14 representative examples, E_R and 81 representative sublanguage examples for generalization, E_σ. The learned grammar has 14 grammar rules and 14 learned compositional constraints. Appendix A.1.1 gives the full set of representative examples, all the grammar rules, and sample compositional constraints. The last column of Table 7.2 shows the categories (nonterminals) used in this grammar. We have 5 nonterminals for auxiliaries and 1 nonterminal for subject. We introduced the subject at this level in order to facilitate agreement and subject-auxiliary inversion that appears in questions. In this grammar, we only have pronouns and proper names as subject. More complex subject constructions will be in-

[1] E_σ is used as performance criteria for generalization, and thus it is parsed at every generalization step of each rule in order to choose the rule that parses the biggest number of examples from E_σ.

[2] For each group, the system keeps the entire learning history (e.g., Appendix A.2).

[3] E_σ needs to be conformal.

troduced incrementally at different group levels (e.g., when grammar rules for noun phrases are learned). Grammar rules for the nonterminal (category) avo model simple forms of auxiliaries *be* and *have* as well as modal auxiliaries and the periphrastic auxiliary *do*, together with subject agreement and inversion. Also at this level we model constructions with relative pronouns used either in questions or relative clause constructions, where we do not have inversion. The category av1 introduces negation, av2 introduces future tense, av3 models 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 ....*)

2. **non-finite auxiliary verbs.** To learn non-finite forms of auxiliaries, we used 5 representative examples and 12 representative sublanguage examples. The learned grammar has two categories nav0 and nav1, for simple and complex non-finite auxiliaries, respectively. We learned 5 grammar rules and 5 compositional constraints.

3. **non-finite verbs.** Using the already learned grammar of auxiliary as background knowledge, we learned a grammar for nonfinite verbs. At this level we have five categories nv, nvo, snvo, obj, pobj. Learning has been done on 5 smaller groups: 3 for nv, 1 for nvo, and 1 for snvo, respectively. The category nv models nonfinite verbs alone (e.g., *to love, loving, having loved*), nvo models verbs and their object(s) (e.g., *loving me, to give me something*), and snvo models subject+verb+object(s) (*him to love me*). The category obj is just a proper noun or a pronoun. As in the case of subj, more complex object will be introduced at later stages. Overall, we used 35 representative examples

and 88 representative sublanguage examples. The learned grammar contains 35 grammar rules and 35 compositional constraints.

4. **finite verbs.** In the next step, we learned grammar rules for finite verbs, again using the already learned grammar as background knowledge. The grammar covers 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 28 representative examples, E_R, 115 representative sublanguage examples. We have two categories sv and svo. The learning has been done on 4 smaller groups, in order to reduce the size of E_σ at each learning stage to increase time efficiency (we have 3 groups for sv, and 1 group for svo). At the sv level we only have constructions of subject-verb, while at svo we have constructions including subject, verb, and object(s). The svo constructions represent the verb together with all its complements (objects), but no adjuncts. The adjuncts will be introduced at the clause level. In this learning stage, we did a coverage experiment for complex finite verbal constructions, because they are highly regular and we could manually build an exhaustive unannotated test set. At this stage subj, obj, pobj are only pronouns and proper nouns, and thus the complexity of the noun phrases does not interfere. The test set has been fully covered and correctly parsed and generated back using the reversible robust parser. We manually validated the output representation (OntoSeR) of each utterance.

5. **noun phrases.** The grammar learned in this group models several phenomena of complex noun phrases (n), either premodified by adjectives (a), genitive

Sample of learned grammar for noun phrases (5)
$\mathrm{a}(\binom{h}{b}) \rightarrow \mathrm{adj}(\binom{h_1}{b_1})$, $\Phi_{comp83}(h, h_1)$, $\Phi_{onto}(b)$.
$\mathrm{a}(\binom{h}{b}) \rightarrow \mathrm{adv}(\binom{h_1}{b_1})$, $\mathrm{a}(\binom{h_2}{b_2})$, $\Phi_{comp84}(h, h_1, h_2)$, $\Phi_{onto}(b)$.
$\mathrm{a}(\binom{h}{b}) \rightarrow \mathrm{a}(\binom{h_1}{b_1})$, $\mathrm{coord}(\binom{h_2}{b_2})$, $\mathrm{a}(\binom{h_3}{b_3})$, $\Phi_{comp85}(h, h_1, h_2, h_3)$, $\Phi_{onto}(b)$.
$\mathrm{na}(\binom{h}{b}) \rightarrow \mathrm{noun}(\binom{h_1}{b_1})$, $\Phi_{comp86}(h, h_1)$, $\Phi_{onto}(b)$.
$\mathrm{na}(\binom{h}{b}) \rightarrow \mathrm{na}(\binom{h_1}{b_1})$, $\mathrm{na}(\binom{h_2}{b_2})$, $\Phi_{comp87}(h, h_1, h_2)$, $\Phi_{onto}(b)$.
$\mathrm{nc}(\binom{h}{b}) \rightarrow \mathrm{noun}(\binom{h_1}{b_1})$, $\Phi_{comp88}(h, h_1)$, $\Phi_{onto}(b)$.
$\mathrm{nc}(\binom{h}{b}) \rightarrow \mathrm{na}(\binom{h_1}{b_1})$, $\mathrm{nc}(\binom{h_2}{b_2})$, $\Phi_{comp89}(h, h_1, h_2)$, $\Phi_{onto}(b)$.
$\mathrm{n}(\binom{h}{b}) \rightarrow \mathrm{nc}(\binom{h_1}{b_1})$, $\Phi_{comp90}(h, h_1)$, $\Phi_{onto}(b)$.
$\mathrm{n}(\binom{h}{b}) \rightarrow \mathrm{a}(\binom{h_1}{b_1})$, $\mathrm{n}(\binom{h_2}{b_2})$, $\Phi_{comp91}(h, h_1, h_2)$, $\Phi_{onto}(b)$.
$\mathrm{n}(\binom{h}{b}) \rightarrow \mathrm{det}(\binom{h_1}{b_1})$, $\mathrm{n}(\binom{h_2}{b_2})$, $\Phi_{comp92}(h, h_1, h_2)$, $\Phi_{onto}(b)$.
$\mathrm{n}(\binom{h}{b}) \rightarrow \mathrm{pn}(\binom{h_1}{b_1})$, $\Phi_{comp93}(h, h_1)$, $\Phi_{onto}(b)$.
$\mathrm{n}(\binom{h}{b}) \rightarrow \mathrm{pro}(\binom{h_1}{b_1})$, $\Phi_{comp94}(h, h_1)$, $\Phi_{onto}(b)$.
$\mathrm{sbj}(\binom{h}{b}) \rightarrow \mathrm{n}(\binom{h_1}{b_1})$, $\Phi_{comp106}(h, h_1)$, $\Phi_{onto}(b)$.

Sample of learned constraints for noun phrases (5)	
$\Phi_{comp87}(h, h_1, h_2) =$	$\{h$.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$\}$
$\Phi_{comp90}(h, h_1) =$	$\{h$.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$\}$
$\Phi_{comp92}(h, h_1, h_2) =$	$\{h$.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.mod, h.head=h_2.head, h_1.cat=det, h_2.cat=n, h_2.det=no$\}$

Figure 7.1: Samples of learned grammar rules and constraints for noun phrases

constructions (d), or postmodified by prepositional phrases (p, pc). We also model noun-noun compounds (na, nc), and have coordination among all the categories (n, a, pc, d). At this stage sbj, obj and pobj are refined. We used 26 representative examples and 53 representative sublanguage examples.

In Figure 7.1, we give a fragment of our learned grammar for noun phrases and selected compositional constraints.[4] In Appendix A.4.1 and Appendix

[4]We present here the DCG-like form of the rules, where the nonterminals are given by lowercase letters, but the arguments and the constraints are kept in the theoretical notation of LWFG for readability reasons. The actual DCG form for all our experiments is given in Appendices.

A.4.2 we give a sample of the representative examples from which this fragment was learned, and present all the grammar rules and sample constraints, respectively. 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. The compositional constraint is also given (Φ_{comp87} in Figure 7.1). 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: *y* and *no* respectively (otherwise we should have used two nonterminals n1 and n2). The constraint Φ_{comp92} will constrain the rule $\mathsf{n} \rightarrow \mathsf{det\ n}$, and thus it will not generate recursively ungrammatical phrases such as *the the disease*. This is obtained 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 *y*, while for the noun on the right-hand side is *no*.

6. **finite and non-finite clauses.** Using the already learned grammar as background knowledge, at this stage we learned rules for nonfinite clauses (ncl), which include the to-nonfinite clauses (tocl), and finite clauses (fcl). At this stage we introduce adjuncts adjc, which in our case are adverbs and prepo-

sitional phrases. At this level are learned the grammar rules for raising and control constructions, which we have described extensively both from a syntactic and semantic point of view in Section 6.2. To learn this grammar fragment, we use 16 representative examples and 41 representative sublanguage examples. In Appendix A.4.1 and Appendix A.4.2, we give samples of representative examples, learned grammar rules and learned compositional constraints for raising and control constructions.

7. **relative clauses.** Next we learned grammar rules for wh-relative clauses and that-relative clauses. We used 10 representative examples and 25 representative sublanguage examples. At this stage we also learned a rule for n, which represents noun phrases postmodified by relative clauses. In Section 6.3.1 we have given examples of relative clauses covered by our grammar.

8. **wh-questions.** In order to learn wh-questions we used 7 representative examples and 16 representative sublanguage examples. As we have mentioned in Section 6.3.3 and Figure 6.26, pg. 182, we used resumptive pronouns and nouns for learning wh-questions. The long-distance filler-gap dependencies are treated by the robust parser. An example is given in Appendix A.5. A discussion of wh-questions, and filler-gap dependencies was given in Sections 6.3.2 and 6.3.3.

9. **copula *to be*.** We learned grammar rules for copula *to be* in order to account for our definitional sentences. We introduced two new categories npred for nominal predicate, and fcl1 for the finite clause that contains copula *to be*, which as we treat as an auxiliary verb. As we mentioned in Section 8.1.1, in

this experiment we only interpret the *predicative-be*. We used 4 representative examples and 6 representative sublanguage examples for learning.

10. **reduced relative clauses.** The grammar rules for reduced relative clauses rrcl and noun phrases postmodified by reduced relative clauses n have been learned from 6 representative examples and 11 representative sublanguage examples. In Section 6.3.1 we have presented examples of reduced relative clauses, and in Appendix A.4.1 and Appendix A.4.2 we give samples of representative examples, learned grammar rules and learned compositional constraints.

This experimental grammar has been tested incrementally on each group of phenomena, based on a benchmark of 823 grammatical examples we developed. We manually validated the output of the parser/generator for each of these examples. This test was not intended to evaluate coverage in the broad-coverage sense, but to test the correctness of our learned grammar, with respect to the phenomena under study.

From the above description and from Table 7.2 we see the practical advantage of our framework: 1) the use of background knowledge (previously learned grammars are added to the background knowledge); 2) the implementation of sound merging and revision methods. At later stages, some rules are added that subsume the previously learned rules, and thus a revision is provided. This is the case when for noun phrases we learned a rule for sbj $\rightarrow$ n, which subsumes the already existing rules defined for auxiliaries: sbj $\rightarrow$ pro (pronoun), and sbj $\rightarrow$ pn (proper noun), respectively. This learning methodology, which guarantees sound grammar revision, allows us to learn additional phenomena at any time. In our case, due to our defi-

nitional corpus requirement, we had to learn rules for copula *to be* and for reduced relative clauses. For the latter, a new rule for n has been added to account for the fact that noun phrases can be postmodified by reduced relative clauses. Table 7.2 shows a summary of all the categories (nonterminals) for which rules are learned in each group. We can also see that the number of learned grammar rules and learned compositional constraints equals the number of representative examples, i.e., 151.

In Chapter 3 we have presented Algorithm 1 which determines whether a Context-Free Grammar is a Well-Founded Grammar, and provides the partial order among all the nonterminals of a grammar (including preterminals). In Figure 7.2 we show the graphical output of this algorithm for our learned grammar. The nodes are the nonterminals and the directed arcs show the partial order $\succ$. The self-loops indicate recursive rules, while the dotted arcs show the inverse relation $\prec$ introduced by non-ordered rules. From this figure we can see that this learned experimental grammar has all type of rules we theoretically mentioned in the definition of a LWFG in Chapter 3. That is, we have ordered non-recursive, ordered recursive, and non-ordered grammar rules.

In order to avoid overgeneralization we have used both chains of nonterminals, and discriminative attribute, as was theoretically mentioned in Section 5.2 (Figure 5.6, pg. 107). For auxiliary verbs for examples, we have a *linear chain* {av4, av3, av2, av1, av0}. We also have *crossing chains* (see Section 5.3.1.4, Figure 5.19, pg. 127): {relcl, fcl, svo, sv} and {whcl, fcl, svo, sv}, where fcl is the crossing point. We also experimented use *discriminative attributes* in order to avoid introduction of a new category (nonterminal). We presented an example at the noun phrase level, for the rule n $\rightarrow$ det n and the constraint Φ_{comp92}, where we used a feature

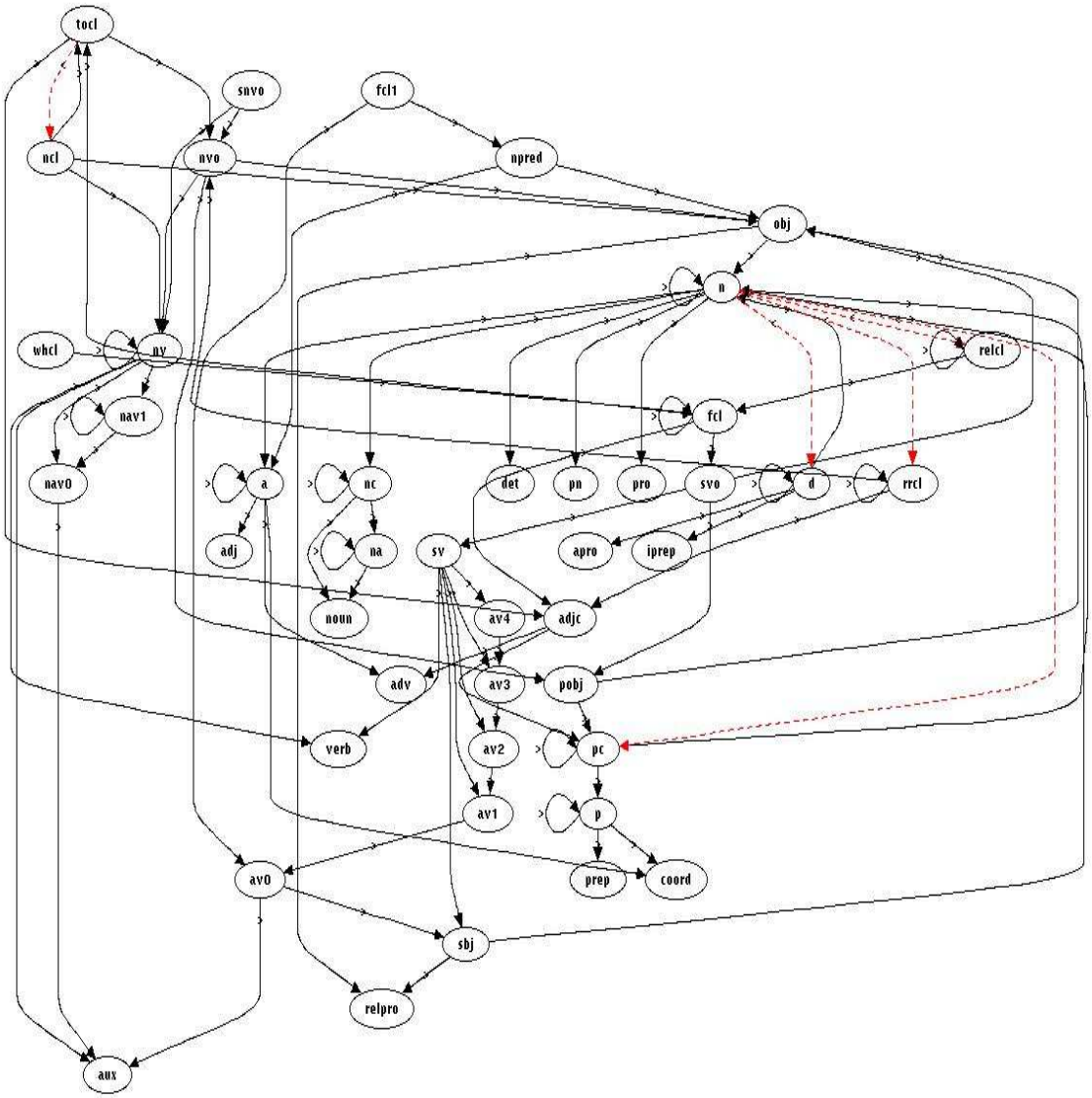


Figure 7.2: Partial ordering relation among nonterminals

attribute det.

An intuitive idea of incremental learning is to learn phenomena that are incrementally more complex. Thus, it seems intuitive to learn first auxiliary constructions, then finite verbs, then clauses and so on. Analyzing the attributes required to annotate the representative examples, we can see that for the lower-level categories (e.g., av, sv) we have more attributes, and the feature attributes are more directly connected to lexical features (pers, nr, etc.), while the higher level categories (e.g., relcl) do not have as many attributes, and the feature attributes are more functional in natures (e.g., stype, ctype for subject and complement type, respectively). Thus, a category can "delete" attributes or "add" new attributes. Also we can have attributes that are carried underspecified by the lower level categories, and they get instantiated at the appropriate category. An example is the attribute int (interrogative), which remains underspecified till it gets either to the category relcl when it becomes no, or to the category whcl when it becomes y (see Appendix A.3.1).

7.1.3 Controlled Experiments

In the previous section, we described the grammar that we have learned for different linguistic phenomena, and the incremental process of learning.

In this section we present additional experiments used for learning this grammar in order to show the practical aspects of all our theoretical algorithms.

Ordered vs. Unordered Representative Examples. In Section 5.1, we provided two algorithms for learning LWFGs. Algorithm 7 assumes an ordered set of representative examples and provides efficient learning, having the hypothesis

search space as a Boolean Algebra (see Section 5.3.1.3). Algorithm 9 is an iterative algorithm which allows us to learn from an unordered set of representative examples, having the search space a complete grammar lattice (see Section 5.3.1.2). The practical advantage of the iterative algorithm is that sometimes when modeling complex linguistic phenomena we might not always know the exact correct order of the examples. The theoretical proof of the search space properties guarantees that both grammar learning algorithms converge to the same grammar. Thus, they can both be used and the resulting learned grammar is the same. In Appendix A.1 we give an example of using both of these algorithms to learn the grammar of finite auxiliaries. For the iterative algorithm, we choose a random order of representative examples (Appendix A.1.2). For this representative example set, we needed four iteration steps for convergence. We can see that the grammar learned from the unordered examples is the same as the grammar learned from the ordered examples.

How do we know that E_R is the representative set? We theoretically define the representative example set as being the simplest set of examples which can be derived from a LWFG and which covers all the grammar rules (Section 3.6, pg. 62). When choosing a representative example set in practice, we are not sure if this is the exact representative set, that is, we may not have chosen the simplest examples. In Section 3.6 we have provided Algorithm 4, which, given a grammar G, and a sublanguage E_σ, generates the representative examples of that grammar (annotated with the semantic molecules). Using this algorithm, we can check if indeed our representative example set was representative. A run of this algorithm for finite auxiliary verbs is given in Appendix A.6.

How do we know that E_σ is conformal? If the representative sublanguage E_σ is not conformal w.r.t a grammar, we do not reach the top of the complete grammar lattice, the learned grammar being one of the grammars of the lattice. In practice, we can detect this by the final coverage of E_σ. If an example is missing, the generalization cannot be done, and thus some of the examples remain uncovered.[5] Another reason is that examples can be wrongly annotated. We can detect some of these mistakes in annotation by taking advantage of grammar reversibility. At the end of each learning iteration, the entire E_σ set is parsed and then generated back. In the next paragraph we say more about the parsing/generation control.

The role of the robust parser/generator. We have extensively taken advantage of our reversible parser/generator throughout our experiments. Reversibility has been used to signal over/under generalization and nondeterminism. This has sometimes led us to refine our annotation of the representative examples. In an initial experiment of learning relative clauses, we have not taken into account the agreement between the modified noun and the embedded verb. In that case, for the utterance *the boy who loves Mary tries to be loved by her*, we obtained only one parse, but when using that representation as input for the generator we obtained back 8 utterances given below:

1. the boy who love mary does try to be loved by her
2. the boy who love mary tries to be loved by her
3. the boy who loves mary does try to be loved by her
4. the boy who loves mary tries to be loved by her
5. the boy who do love mary does try to be loved by her
6. the boy who do love mary tries to be loved by her
7. the boy who does love mary does try to be loved by her
8. the boy who does love mary tries to be loved by her

[5] This can be noticed from the learning history which allows the validation of our learning mechanism, thus acting as learning explanation.

From this output we can clearly see that agreement is not done. This control enabled us to revise the representative examples in order to account for agreement, and finally to obtain only the valid generated sentences (3,4,7,8). In Appendix A.3.2 and Appendix A.3.3 we give the full runs of the parsing/generation example for the relative clauses with and without agreement, respectively. In Appendix A.3.1 we give an example of parsing/generation for the clause *who can not have been going* which has 5 parsing outputs due to different categories — sv, svo, fcl, relcl, whcl — even if the semantic representation OntoSeR, is the same. For generation, we obtained three utterances corresponding to this OntoSeR due to lexical variation of *cannot*, *can't*, and *can not*. In Appendix A.3.4 we give a more complex utterance *King Abdullah has been working to try to get the Palestinian leaders to come to the table*, for which we obtain only one parse that is generated back as one utterance.

7.2 Concluding Remarks

From these experiments we have seen that the learned experimental LWFG (151 rules and 151 compositional constraints) is complex, it covers diverse linguistic phenomena, it can be incrementally extended, and it can be used for deep semantic analysis (see Chapter 6). The refinement of any grammatical aspects involves only the refinement of the elementary semantic molecules and of the representative examples/sublanguage, and not the refinement of the grammar rules and compositional constraints. This important feature is possible due to our sound grammar revision and merging methods. The results of this section and the qualitative evaluation in Chapter 6 allow us to draw the following conclusions:

- The representative examples, E_R, and the representative sublanguage E_σ can be built based on linguistic knowledge about categories and their specific attributes.

- The learning algorithms presented theoretically in Part I, allow us to verify the E_R/E_σ adequacy and consistency, and allow grammar revision for incrementally building an experimental grammar. The learning can be done both from ordered and unordered representative examples. The theoretical properties of the search space (boolean algebra and complete grammar lattice, respectively) guarantee that the learned grammar is the same regardless of the order of the examples.

- The learned experimental grammar covers complex linguistic phenomena (e.g, raising and control, long-distance dependencies, noun compounds, nominalization, complex verbs with auxiliaries, complex noun phrases)

- Our semantic representation OntoSeR is expressive enough to be able to represent these linguistic phenomena and it is useful for deep language understanding (concept level answers to questions)

In Chapter 8 we will show that OntoSeR together with the *weak concept identity principle* is useful for direct knowledge acquisition. We present our semantic interpreter which implements this principle, and we show, in a pilot experiment, that we can build semi-automatically OKR-annotated treebanks (human validation is required to choose just the correct annotation). This type of treebanks can be used in future research to further develop the semantic interpreter towards a strong semantic context.

Chapter 8

OntoSeR for Knowledge Acquisition

In Chapter 6 we have shown the expressiveness of our semantic representation, OntoSeR with respect to diverse and complex linguistic phenomena covered by our learned grammar, as well as its suitability for semantic role labeling and straightforward handling of questions and answers. As we have mentioned before, we define the meaning of an utterance as the set of answers given to all the questions that can be asked with respect to that utterance. We come back to this issue in Section 8.3.

In this chapter, we present the usefulness of OntoSeR in direct knowledge acquisition, with focus on terminological knowledge. For this, we first present a more refined definition of the OKR and a particular treatment of the concept identity principle adequate to terminological knowledge.

The OKR for terminological knowledge is a directed acyclic graph (DAG), $G = (V, E)$, where the vertex set V is a set of concepts and instances of concepts, and the set of edges E has associated a labeling function $sr\colon E \rightarrow S_R$, where S_R is

a set of semantic roles.[1] This representation is similar to conceptual graphs (Sowa, 1999), except that the roles are labeled edges in DAG, and not vertices as in Sowa's conceptual graphs. In Figure 8.1 we show an example of OKR for terminological knowledge in geometry domain obtained from definitions given in natural language (see more details on the acquisition process in Section 8.1.1). This example will be used to show the main theoretical concepts introduced in this chapter.

We denote the concepts in OKR by *#name_concept* (e.g., #square in Figure 8.1). The concepts constitute a hierarchy of concepts, based on the subsume relation (sub), which is the inverse of the is_a relation. In this dissertation we denote by #concept the top element of this hierarchy. This is the only concept that has a self-loop. An instance of a concept is denoted by the name of a concept followed by the instance number (e.g., #angle7). A concept and the instance of this concept are distinct vertices in OKR, having the same name (e.g., $name(\#angle) = name(\#angle7) = angle$).

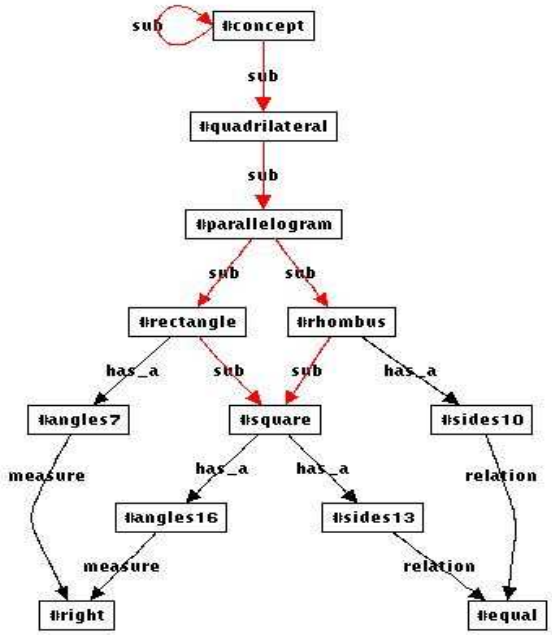
The semantic role which labels an edge $(u, v) \in E$ is denoted by $sr(u, v)$, and the set of vertices adjacent to a vertex $u \in V$ is $adj[u] = \{v \in V : (u, v) \in E\}$.

A DAG is called *rooted* at a vertex $u \in V$, if there exists a path from u to each vertex of the DAG. Let us consider an OKR (i.e., DAG) rooted at #concept, and let us denote by $subDAG[u]$ a subDAG rooted at the vertex $u \in V$. We have the following definition:

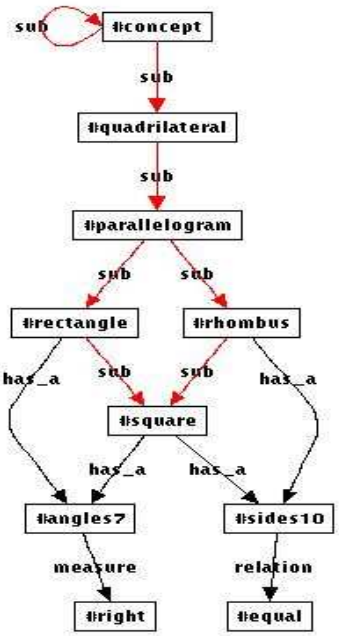
Definition 34. *Two subDAGs rooted at two vertices u, u' are equal if the set of the adjacent vertices to u and u' respectively, are equal and if the edges incident from u and u' have the same semantic roles as labels. That is:*

$subDAG[u] = subDAG[u'] \stackrel{def}{=} adj[u] = adj[u'] \wedge \forall v \in adj[u],\ sr(u, v) = sr(u', v)$

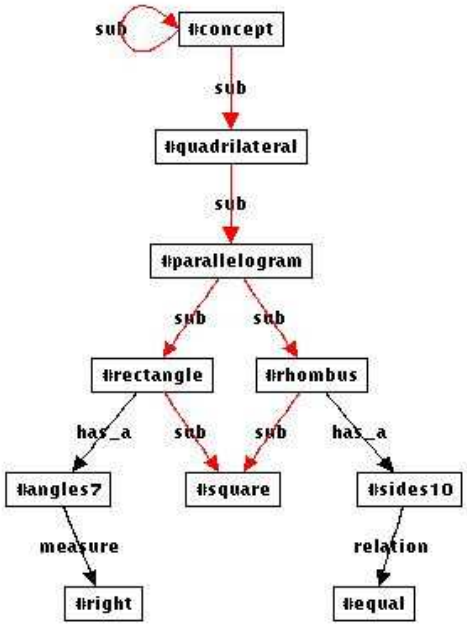
[1] In the remainder of this chapter, when we say OKR we refer to OKR for terminological knowledge.



(a) initial OKR



(b) P5 satisfied



(c) P5+P6 satisfied

Figure 8.1: Concept identity for terminological knowledge

Property 5 (P5). In an OKR, all vertices $u, u' \in V$ with the same name, and whose subDAGs are equal are identical (i.e., the same vertex in OKR). That is:

$$\forall u, u' \in V, name(u) = name(u') \wedge subDAG(u) = subDAG(u') \Rightarrow u = u'$$

Using a hash table, there is a linear algorithm $O(|V| + |E|)$ which transforms an OKR to an equivalent OKR which satisfies Property 5. The algorithm is similar to the algorithm which finds the common subexpressions of a given expression.[2]

In Figure 8.1(a) we can see that the vertices #angle7 and #angle16 have the same name and their subDAGs are equal. The same is true for the vertices #sides10 and #sides13. Using the above mentioned algorithm the initial OKR in Figure 8.1(a) is transformed to an equivalent OKR, which satisfies Property 5, shown in Figure 8.1(b).

Property 6 (P6). A concept in a hierarchy of concepts can be linked by the sub relation only to its parent(s), and not to any other ancestors. A subDAG defining a property of a concept from the hierarchy of concepts can be found only once in the OKR at the level of the most general concept that has this property.

The OKR in Figure 8.1(b), does not satisfy Property 6, since the subDAG corresponding to the property "right angles" for example, belongs both to the concept #rectangle and its child #square. Similarly, the subDAG corresponding to the property "equal sides" belongs both to the concept #rhombus and to its child #square. By applying Property 6, we obtain the OKR in Figure 8.1(c). In this way, we can see that the properties "right angle" and "equal sides" belongs directly only to #rectangle and #rhombus, respectively, and not to #square, which is lower in the hierarchy. In this way we remove redundancy, and implement the notion of inheritance. Thereby, the concept #square inherits the properties "right angles" and "equal sides".

[2]Property P5 guarantees that the OKR is a DAG with totally shared subDAgs.

Principle 7 (Weak Concept Identity for Terminological Knowledge). *An OKR which satisfies Property 5 and 6 is an OKR which satisfies the weak concept identity principle.*

The OKR in Figure 8.1(c) is an example of OKR which satisfies the weak concept identity principle.

Our semantic interpreter implements the weak concept identity principle for terminological knowledge. Acquiring terminological knowledge from natural language requires a proper treatment of the semantics of copula *to be*, as can be seen from the examples (1). All the examples given in this chapter are covered by our learned grammar.

(1) Acne is a skin disease.

Acne is very common.

Acne is an inflammatory skin disease characterized by pimples.

Acne is an inflammatory disease involving the sebaceous glands of the skin.

In Section 8.1 we present the treatment of copula *to be* in our framework. Section 8.2 presents an application of acquiring terminological knowledge from natural language definitions in the medical domain. In Section 8.3 we discuss the concept of meaning as answers to questions from a practical point of view (this concept was formally defined in Chapter 4). In Section 8.4 we describe a pilot experiment of building an OKR-annotated treebank from medical definitions and querying it using natural language (wh-questions).

8.1 Copula *to be*

The verb *be* as used in sentences (2) is referred as the copula. It can be followed by a predicative phrase (a noun phrase, or an adjective).[3]

(2) a. Pat is a graduate student.

b. Pat is smart.

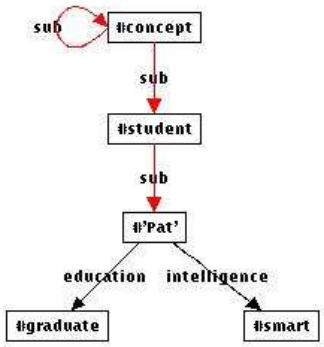
As stated in many linguistic theories, the syntactic behavior of the copula follows the auxiliary verbs rather than the main verbs (Gazdar et al., 1985; Pollard and Sag, 1994; Joshi and Schabes, 1997). We take the same approach in our framework, considering it as an auxiliary verb. However, copula is not followed by a verbal category (by definition) and therefore it must be the rightmost verb. In this respect it behaves like a main verb. In our framework we learn a special grammar rule for the copula *to be*.

The semantic behavior of the copula is also unlike the main verbs. Any semantic restrictions or roles placed on the subject come from the complement phrase, rather than from the verb. Because the complements predicate over the subject, these types of sentences are often called *predicative* sentences, and the copula *to be* is called *predicative-be*. In Figure 8.2 we show the OKR of the sentences given in (2). We can see that all the properties are attached to the vertex corresponding to the subject, i.e., *Pat*.

A fundamental semantic property of the utterances involving predicative-be is that the subject is the most specific concept (or, an instance of a concept, if we have factual knowledge), while the complement is the more general concept. Thus,

[3]In this dissertation we are not concerned with the predicative PP.

Student is a concept.[a]
Pat is a graduate student
Pat is smart



[a]The reader should keep in mind that #concept is a generic name we have assigned to the top element of the hierarchy of concepts. The hierarchy can be extended to have many top elements similar to WordNet(Miller, 1990), and other names can be chosen for this top concept.

Figure 8.2: Predicative-be

the sentence in (2a) cannot have the complement and the subject inverted, i.e., *A graduate student is Pat*, since Pat is an individual and *a graduate student* is a more general concept.[4]

The copula *be* can be used in examples where this sort of inversion between the subject and complement can occur, as can be see in (3). We have in this case the *equative-be*.

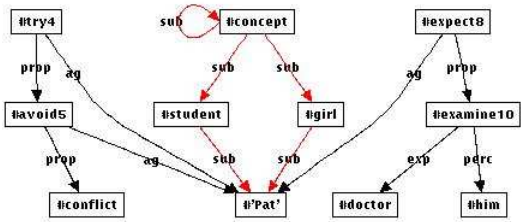
(3) a. Pat is the student who tried to avoid conflict.

b. The girl who expects him to be examined by the doctor is Pat.

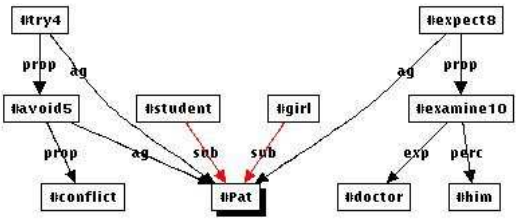
c. The girl who expects him to be examined by the doctor is the student who tries to avoid conflict.

[4]The predicative inversion in English is irrelevant as it is purely syntactic.

Student is a concept
Girl is a concept
Pat is the student who tried to avoid conflict
The girl who expects him to be examined by the doctor is Pat

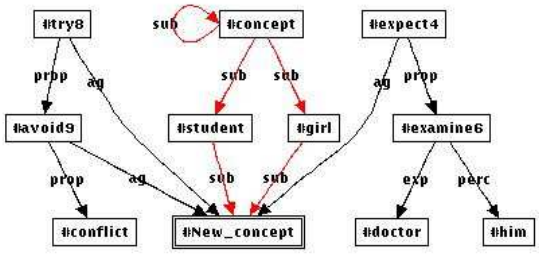


Question: Who is Pat?
Answer: #'Pat'



(a)

The girl who expects him to be examined by the doctor is the student who tries to avoid conflict



(b)

Figure 8.3: Equative-be

Equative constructions can be used only if both the subject and the complement of *be* are individuals (given either as a proper name of an individual in a factual KB, or as a fully determinate instance of a concept).

In the examples (3a-b), we have an individual given by its proper name, and an individual given as a determinate concept. In Figure 8.3(a) we give the OKR for the sentences in (3a-b) as well as the answer to the question *Who is Pat?*. In (3c) we have a different equative construction, in which the name of the concept is not given. In this case we have a "New_concept" which will have the properties of both

the left and the right concept. Comparing Figure 8.3(a) and 8.3(b) we can see that if we add the statement *This is Pat* after (3c) we will have the same representation, having identified the name of the concept as being *Pat*.

8.1.1 Copula *be* for terminological knowledge

We have seen that copula *be* can be either predicative or equative. For acquiring terminological knowledge from natural language statements, however, we only consider the *predicative-be*. This assumption holds even if we have natural language definitions as a source of terminology, as can be seen in (1) and discussed also in more detail in Section 8.2. The intuition is that the equivalence between the *definiendum* and the *definiens* is a theoretical property, and the definitions we encounter in natural language text and even in dictionaries lack this property. Thus, we cannot consider equative-be in a practical application for acquiring terminology from natural language.

Since we consider predicative-be, the left concept (i.e., subject) is more specific than the right concept (i.e., the complement): e.g., *the quadrilateral with equal sides is a polygon with equal sides*. We can also have the predicative phrase given by an adjectival phrase.

In the remainder of this section, we present an illustrative example for acquiring terminological knowledge from natural language definitions, in the geometry domain. This example emphasizes both the theoretical concepts introduced at the beginning of this chapter, and the semantics of predicative-be. Let us consider the definitions below:[5]

[5]As in generative ontology (Jensen and Nilsson, 2003), we ignore the determiners at the OKR level for terminological knowledge (see Chapter 4, footnote 7). Thus, we make no difference among *a concept*, *the concept*, *every concept*, at OKR level for terminological knowledge. In the case of

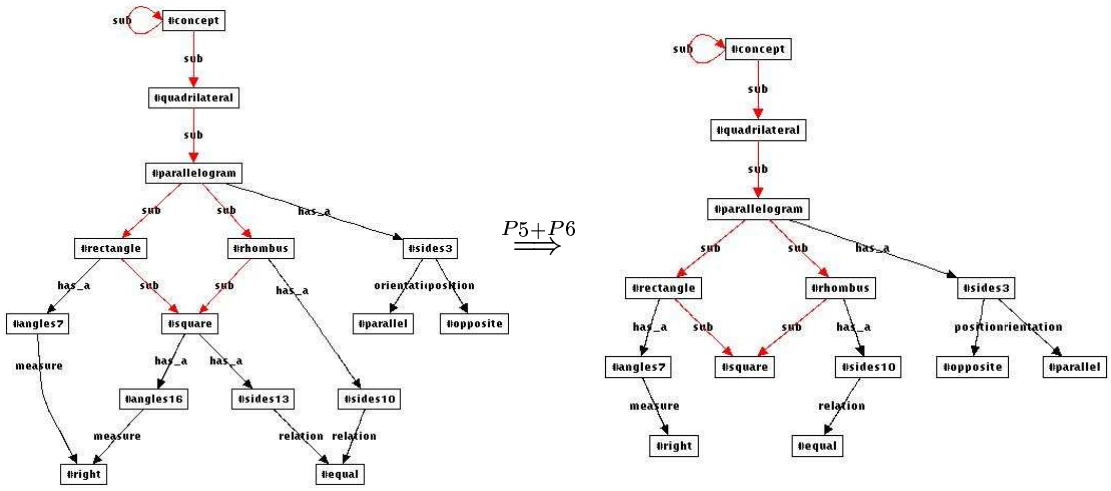


Figure 8.4: Terminological knowledge acquired from the NL definitions in (4)

(4) A quadrilateral is a concept.

A parallelogram is a quadrilateral with opposite parallel sides.

A rectangle is a parallelogram with right angles.

A rhombus is a parallelogram with equal sides.

A square is a rectangle with equal sides.

A square is a rhombus with right angles.

As we mentioned in the introduction of this chapter, we have #concept as the top of the terminological OKR (i.e., the root of the directed acyclic graph). In order to build the sub hierarchy we need to define the concepts: either directly (e.g., *a quadrilateral is a concept*), or indirectly by definitions based on previously introduced concepts (e.g., *a parallelogram is a quadrilateral with opposite parallel sides*). We can also have concepts that have two different definitions that are conceptually different. An example is the concept #square, defined either as *a*

factual knowledge, it is clear that we do need to interpret the determiners. The determiners are represented at the OntoSeR level and they could be interpreted; we leave this for future work.

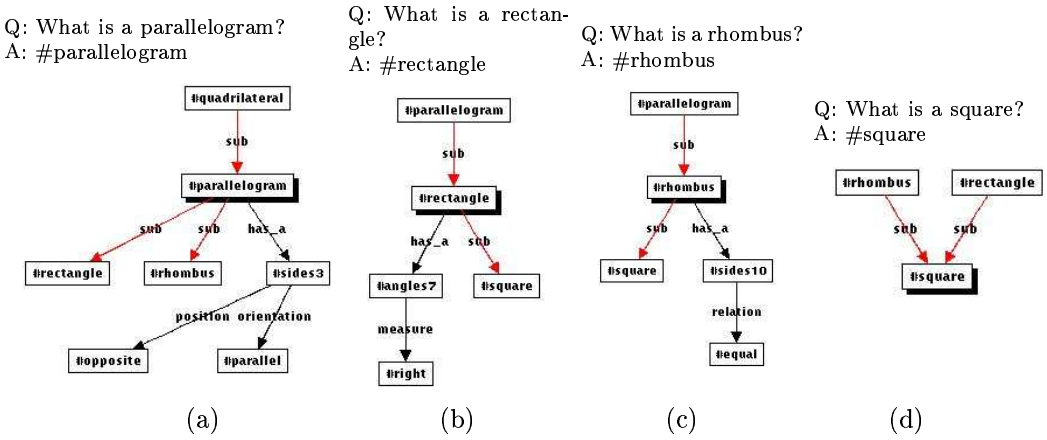


Figure 8.5: NL-querying of the terminological knowledge acquired from (4)

rectangle with equal sides or *a rhombus with right angles.* These types of definitions lead to the issue of multiple inheritance present at the level of OKR.

In Figure 8.4 we give the OKR that was obtained from the NL definitions in (4). The OKR on the left, does not satisfy Properties 5 and 6 (we have redundant attributes for the concept #square). The OKR on the right satisfies these two properties, i.e., the weak concept identity principle for terminological knowledge.

After the OKR is built we can query it using natural language. Examples of such NL-queries are given in Figure 8.5, showing the usefulness of our OntoSeR representation as an Ontology Query Language. For example, asking the question *What is a parallelogram?* gives back the concept #parallelogram with all its incident sub edges and all it determinations (subDAGs), i.e., "is a quadrilateral" (sub is the inverse of is_a), "has sides opposite," "has sides parallel," "subsumes rectangle," "subsumes rhombus" (Figure 8.5(a)).[6] As mentioned in Chapter 6 only If we ask

[6]The answers are obtained at the concept level and not generated directly in natural language because at the OKR level we do not have reversibility. As mentioned in Chapter 6, only at the OntoSeR and TKR levels the reversibility of LWFGs can be exploited. Generating natural language strings from OKR would require text generation techniques, which is outside the scope of this dissertation.

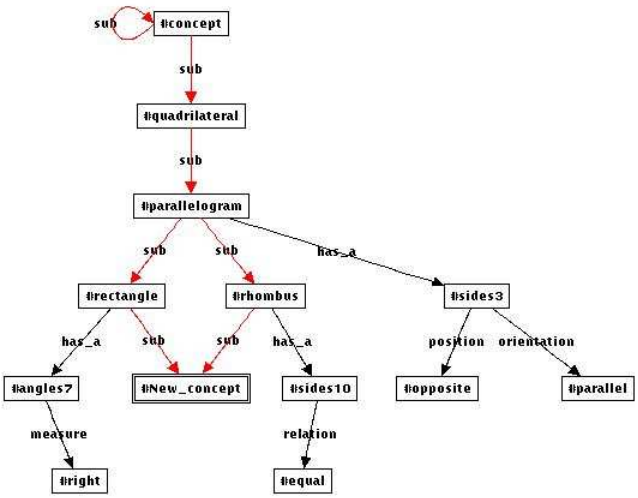


Figure 8.6: Acquisition of nameless new concepts in terminology

What is a square?, we obtain the concept #square with its incident sub edges, i.e., "is a rectangle" and "is a rhombus" (Figure 8.5(d)). As can be seen, due to Property 5 and 6 the concept #square does not have any other determinations (subDAGs), since they can be inherited from its parents. We can thus see that we always obtain conceptual answers, which have a different rationale and outcome than the simple "bag of words" answers extracted directly from the natural language definitions.

In the previous section, we have seen that we can introduce *nameless* new concepts using equative sentences, such as (3c). Since for terminology we only have predicative-be, the *nameless* new concepts are introduced only if we provide statements in both directions, such as (5).

(5) Every rectangle with equal sides is a rhombus with right angles.
Every rhombus with right angles is a rectangle with equal sides.

If we are to use these statements instead of the two definitions of "square" in (4),

we obtain the OKR given in Figure 8.6 which obeys both Property 5 and 6.

8.2 Application: Terminology Acquisition in the Medical Domain

A key source for terminological knowledge is the definition of a term, a fact acknowledged by the abundance of work on processing dictionary-like definitions (Chodorow, Byrd, and Heidorn, 1985; Klavans, Chodorow, and Wacholder, 1992; Wilks, Slator, and Guthrie, 1996; Richardson, Dollan, and Vanderwende, 1998; Moldovan and Rus, 2001; Rus, 2002). However, the definition construction is a notorious problem for terminology work. What exactly is a definition?

A theoretical definition of definitions can be formulated as follows:

Definition 35. *A definition is a syntagma σ, which in the query-form σ_Q has as answer one and only one concept, regardless of further addition of new concepts to the ontology.*

Only in this theoretical view, can we talk about the *logical definition* (Weisman, 1992), which assumes the equivalence between the *definiendum* (term to be defined) and the *definiens* (the definition of the term). Moreover, logical definitions require the definiens to have both a *genus* and a *differentia* (Eck and Meyer, 1995).

In practice, we cannot guarantee neither the theoretical limit, nor the genus-differentia assumption. This practical view is in accordance with studies on scientific dictionaries that show that definitions often contain "encyclopedic information" making the equivalence between the definiendum and definiens questionable (Norman, 2002). As we are focusing on the medical domain, these findings are relevant to our approach.

In this dissertation, we consider as sources of definitions both dictionaries and

definitions extracted from on-line articles. For the latter, we have implemented a system, DEFINDER that mines definitions from on-line consumer-oriented medical articles. We present this system in the next section. Then, we show the characteristics of our definitional corpus, where we can see that the *logical definition* is rather a theoretical concept, not a practical reality.

8.2.1 DEFINDER: Getting NL Definitions from Corpora

A rich resource for definitions is the consumer-oriented medical text. It consists of medical articles or manuals written by specialists for a general audience, where a lot of terminology must be defined. The language of definitions is thus less technical, closer to general vocabulary. We collected a corpus of consumer-oriented medical articles containing over 1M words from different sources on the web.

The acquisition of definitions from textual corpora is a challenging task. The structure of definitions in a text is not always similar to the one in on-line dictionaries. The algorithm for the extraction of definitions from text is a rule-based method implemented in the DEFINDER system (Klavans and Muresan, 2000; Muresan and Klavans, 2002). First, a development set of articles was analyzed and a set of patterns that occur frequently and reliably in many text genres (e.g., articles, book chapters, health newspapers) were identified. We grouped these patterns into three categories: cue-phrases (e.g., *is the medical term for*), text markers (e.g., "–", "(", ")"), and syntactic patterns (e.g. syntactic complement of the copula *to be*, appositional patterns). The identification of definitions from these initial contexts was performed in two steps: 1) shallow parsing for identification of simple definitions and candidate complex definitions, and 2) full parsing of these candidate definitions using a statistical parser (Charniak, 2000).

In the first step, we use Brill's tagger (Brill, 1992) and an NP chunker (Ramshaw and Marcus, 1995) in conjunction with a simple finite-state grammar. We have augmented Brill's tagger lexicon with medical terms from Unified Medical Language System (UMLS) lexicon (Lindberg, Humphreys, and McCray, 1993) to increase accuracy. A filtering step is performed to remove patterns for enumerations, or explanations. As a result of shallow analysis, we can have both $\langle term \rangle (\langle definition \rangle)$ and $\langle definition \rangle (\langle term \rangle)$. When length was not sufficient, a simple statistical measure based on frequency counts was used to discriminate between the term and its definition. This is usually the case for technical/lay pairs, like "tachycardia/irregular heartbeat". In addition, we select candidate definitions that cannot be easily identified by shallow processing and for which full parsing is more reliable.

In the second step, these candidate definitions are syntactically parsed using a statistical parser (Charniak, 2000). We perform a pattern matching over full parse trees in order to identify complex appositives, syntactic complements of the copula *to be* and complex definitions found in the context of text markers. One aspect is worth mentioning here. Even if we have used full syntactic parsers in our definition extraction system, we only used pattern matching of limited depth over the parse trees. Thus, we were able to extract definitions even if the deep tree structure was not correct. While this strategy is sufficient to extract fairly accurate definitions (see Appendix D for DEFINDER's evaluation), it is not good enough if we are to use these parsers to acquire the semantics of these definitions (wrong syntactic parsing will lead inevitably to wrong semantics). The inability of current syntactic parsers, trained on the Wall Street Journal (WSJ) to accurately parse

1. Acne is a skin disease characterized by papules and pustules on the face and neck.

2. Acne is an inflammatory skin disease characterized by pimples that can appear on any part of the body.

3. Acne is a skin disease caused by overactive oil glands.

4. Acne is an inflammatory disease involving the sebaceous glands of the skin. Acne is characterized by papules or pustules or comedones.

Figure 8.7: Multiple definitions of *acne*

medical text is a further motivation for our grammar learning framework. We used samples of the DEFINDER corpus to analyze the characteristics of definitions, in order to learn the linguistic phenomena that characterize them.

In the next section we present the characteristics of our definitional corpus.

8.2.2 Characteristics of the Definitional Corpus

In this section we present briefly the characteristics of our definitional corpus on two dimensions: conceptual and linguistic.

Conceptual Dimension. We consider as our corpus both definitions extracted from corpora, by DEFINDER, and definitions from dictionary-like resources (WordNet glosses). This corpus has the advantage of containing multiple definitions of the same term, which enables us to analyze the conceptual nature of definitions, and to see if the theoretical definition, i.e., logical definition, is the type we find in practice (either in the text, or in existing resources). Figure 8.7 presents multiple definitions of the term *acne*. It can be noticed that each definition has some additional property (e.g., definition 1 specifies the symptoms and their location, while

1. Atherosclerosis is the progressive narrowing of arteries from cholesterol plaque deposits.

2. Hepatitis is a disease caused by infectious or toxic agents and characterized by jaundice, fever and liver enlargement.

3. Hepatitis A is an acute but benign viral hepatitis caused by a virus that does not persist in the blood serum.

4. Hepatitis B is an acute viral hepatitis caused by a virus that tends to persist in the blood serum.

5. Addison's disease is a degenerative disease caused by a deficiency in adrenocortical hormones and characterized by weight loss, brown pigmentation of the skin, and low blood pressure.

Figure 8.8: Sample from the definitional corpus

definition 3 specifies the cause; definition 1 specifies that *acne* is a *skin disease*, while definition 5 specifies that *acne* is an *inflammatory disease*). Thus, neither of these definitions can be considered complete. This incompleteness questions the core assumption of the logical definition: the equation between the definiendum and the definiens. How can we be sure that the equivalence can be obtained? How can we be sure that there is no other definition, which contains an additional property required to fully determine the "acne" concept? Not only that we do not have the equative assumption satisfied, but we do not always have the genus-differentia pattern either. For example, in Figure 8.8 the definition of *atherosclerosis* does not contain the genus phrase, which would be "disease". This conceptual analysis further motivates our choice of interpreting copula *be* as predicative-be instead of equative-be for terminological knowledge (see Section 8.1.1).

Linguistic Dimension. Figure 8.8 gives several definitions from our corpus, illustrating their complex linguistic constructions (both from a syntactic and a semantic

Disease	Body Part	Procedure
cause (287)	carry (58)	use (80)
characterize (73)	locate (41)	remove (42)
mark (62)	lead (18)	destroy (18)
occur (44)	connect (18)	treat (15)
affect (38)	produce (17)	involve (15)
result (32)	contain (16)	help (14)
lead (27)	control (15)	make (13)
produce (25)	cover (13)	cut (12)
form (25)	line (12)	insert (11)
associate (23)	extend (12)	inhibit (10)
destroy (21)	pass (11)	change (10)

Figure 8.9: Top 10 verbs for three semantic types in UMLS

point of view). We have instances of almost all the linguistic phenomena discussed in Chapter 6. We have nominalization (*narrowing of arteries from cholesterol plaque deposits*) and noun compounds (e.g., *cholesterol plaque deposits*, *weight loss*, *blood serum*, *liver enlargement*). We have wh-relative and that-relative clauses (*that does not persist in the blood serum...*), reduced relative clauses (*caused by infectious or toxic agents ...*), and embedded relative clauses (*caused by a virus that tends to persist in the blood serum ...*). One characteristic of the definitional corpus is the presence of prepositions and coordinations. An analysis of our corpus shows that on average there are 3 prepositions per definition. Regarding coordination, definitions 2 and 5 in Figure 8.8 are eloquent examples of the complexity of coordination constructions. We can have coordinations between all categories. While definitions are notorious for their complex noun phrase structure, the verbal constructions are less complex. We have active and passive voice, modals (*can appear on any part of the body*), negation (*does not persist in ...*) and sometimes raising verbs (*tends to persist in ...*), but we do not have complex constructions involving aspect, while the tense is generally the present tense. There is a reduced number of verbs, as

well. We performed a simple frequency count of verbs that appear in definitions based on the semantic types of the definiendum. Table 8.9 shows the top 10 verbs for the semantic types *Disease or Syndrome*, *Body Part, Organ or Organ Component* and *Therapeutic or Preventive Procedure* in UMLS (in Table we use the short names *Disease*, *Body Part*, *Procedure*). In this dissertation, the semantic roles of these verbs have been either adapted from existing general lexical resources (LCS Database (Dorr, 1997)), or manually derived from UMLS (Lindberg, Humphreys, and McCray, 1993).

As we have already mentioned, our learned experimental grammar covers linguistic phenomena that are characteristic for definitional corpus. We need to mention that learning is based on semantically annotated representative examples that were not domain specific, the learning framework being general. Application for the medical corpus could benefit from a stronger semantic context (e.g., domain-specific roles), but this is not a requirement (i.e., we can use generic semantic roles).

8.2.3 Terminological Knowledge Acquisition

The findings from our definitional corpus analysis bear relevance to the process of terminological knowledge acquisition from the definitional text. We saw that we can only have copula *be-predicative*, we cannot assume that there is an equivalence between *definiendum* and *definies*, and we cannot assume that we always have the *genus* phrase.

In this section, we show that our semantic representation, OntoSeR, is adequate for the acquisition of terminological knowledge. We have performed a pilot experiment in order to acquire terminological knowledge, which has been subsequently used for NL-querying. This section constitutes the other side of the *quali-*

2. Hepatitis is a disease caused by infectious or toxic agents and characterized by jaundice, fever and liver enlargement.
3. Hepatitis A is an acute but benign viral hepatitis caused by a virus that does not persist in the blood serum.
4. Hepatitis B is an acute viral hepatitis caused by a virus that tends to persist in the blood serum.

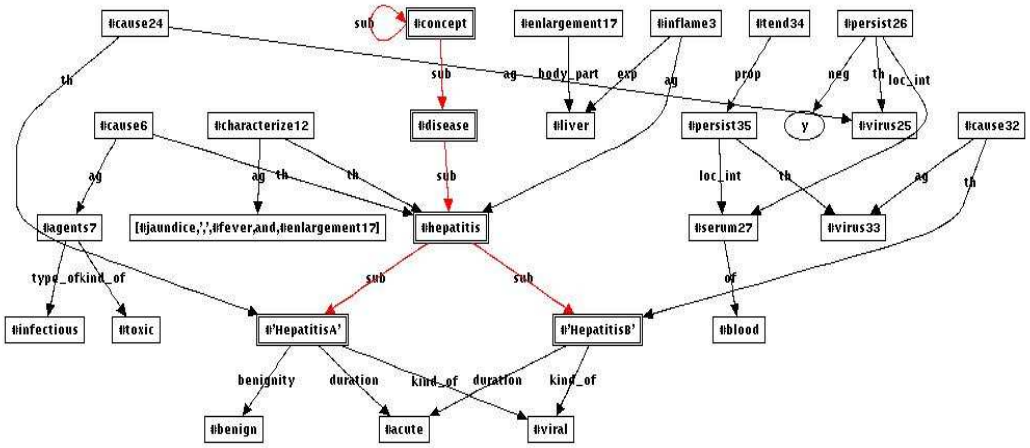


Figure 8.10: OKR for the definitions 2,3,4 given in Figure 8.8

tative evaluation of our semantic representation, OntoSeR, and of our learned grammar (Chapter 6 showed the linguistic expressiveness of OntoSeR). Unlike Moldovan and Rus (2001) and Rus (2002), who use logical formulas for the representation of WordNet glosses, we transform our OntoSeR representation in an OKR form that is a DAG for which we have defined the weak concept identity principle (Principle 7).

In order to show the processes of knowledge acquisition and NL-querying, we present two examples: one of constructing a hierarchy of concepts (definitions 2,3,4 of *hepatitis*, *Hepatitis A* and *Hepatitis B*, from Figure 8.8), and one for merging several definitions of a term (definitions of *acne* given in Figure 8.7). The definitional text and OKRs of these two examples are presented in Figure 8.10 and 8.11, respectively. The questions and answers related to these two OKRs are given in

1. Acne is a skin disease characterized by papules and pustules on the face and neck.
2. Acne is an inflammatory skin disease characterized by pimples that can appear on any part of the body.
3. Acne is a skin disease caused by overactive oil glands.
4. Acne is an inflammatory disease involving the sebaceous glands of the skin. Acne is characterized by papules or pustules or comedones.

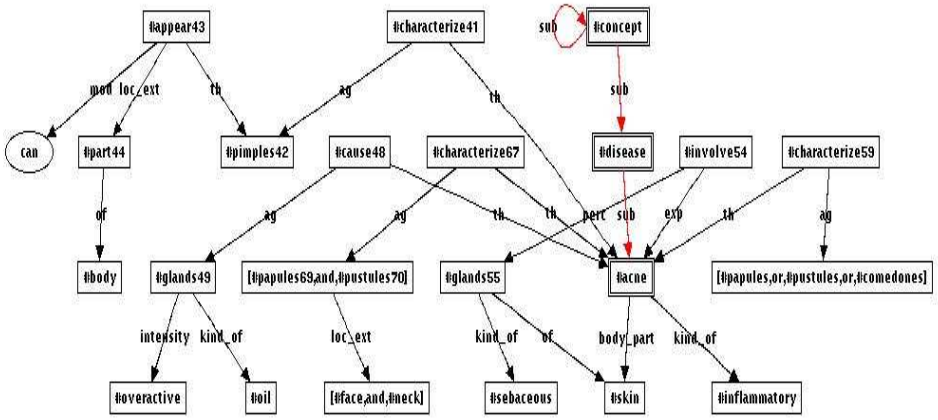


Figure 8.11: OKR for the definitions of *acne* given in Figure 8.7

Figure 8.12 and Figure 8.13, where we give the answer concept(s) together with its defining subDAG(s).

The acquisition of knowledge can be done directly, since we consider both concepts (#hepatitis, #blood) and instances of concepts (#glands49,#glands55, #virus25,#virus33) in our OKR representation (Nirenburg and Raskin, 2004).

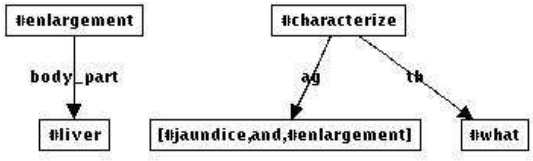
The definiendum is always a concept, and it is part of the sub hierarchy. The concepts in the sub hierarchy are presented with double square boxes in Figure 8.10 and Figure 8.11. All the definitional properties of the concepts are directly linked to the concept vertex (facilitated by our interpretation of copula be-predicative). For example, even if in the text we have *Acne is an inflammatory disease*, the property "inflammatory" is linked to the concept #acne and not to the concept #disease. This is obtained since only #disease was previously part of the sub hierarchy. If the

concept #inflamatory_disease is present, then this most specific concept is selected as the direct parent of #acne. However, the hierarchy of concepts can be dynamically revised as the system acquires more knowledge. If we encounter concepts that are diseases and they have the "inflammatory" property, this information can be collected and a new concept can be proposed to be added in the hierarchy. In this dissertation we are not concerned with this revision, but our representation clearly allows such a step.

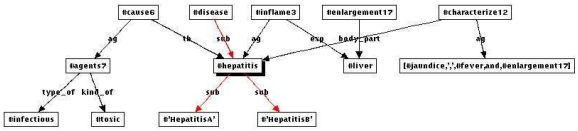
Besides the concepts that are defined, we can also have *concepts that are referred* (i.e., they are part of the definiens), *if they do not have any modification* (e.g., #blood in definition of Hepatitis A , and Hepatitis B, #papules in definition 4 of acne).

If a referred concept has modifications, it is represented as an instance of a concept in OKR. As a consequence, various verbalizations of concept properties can be differentiated in OKR, allowing us to obtain direct answers that are specific to each verbalization. For example, the term *virus* appears in the definition of both *Hepatitis A* and Hepatitis B. In OKR, they are two different instances of a concept, #virus25 and #virus33, since they have different modifications: *persists in the blood serum*, *does not persists in the blood serum*, respectively. These modifications are essential part of the differentia of concepts #HepatitisA and #HepatitisB, making the distinction between the two. When we ask the question *What is caused by a virus that persists in the blood serum?* (Q4 in Figure 8.12), we obtain only the correct answer #HepatitisB (A4 in Figure 8.12). Another example of two different instances of a concept is given by #glands49 and #glands55, in definitions 3 and 4 of *acne* (Figure 8.11). Asking the question *What causes acne?* (Q2 in Figure 8.13),

Q1: What is characterized by jaundice and liver enlargement?

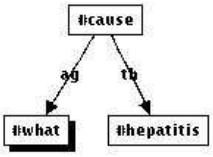


A1: #hepatitis

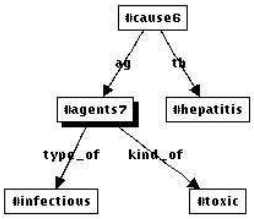


(a1)

Q2: What causes hepatitis?

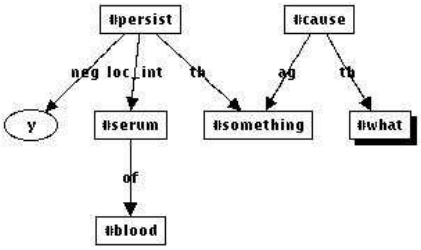


A2: #agents7

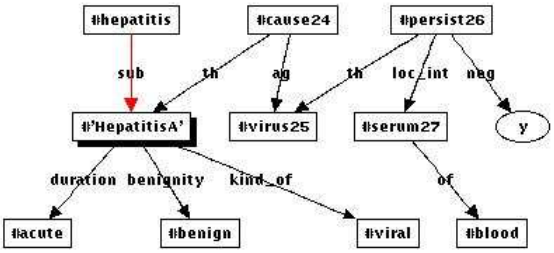


(a2)

Q3: What is caused by **something** that does not persist in the blood serum?

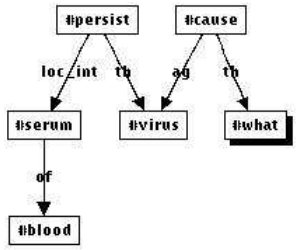


A3: #'HepatitisA'

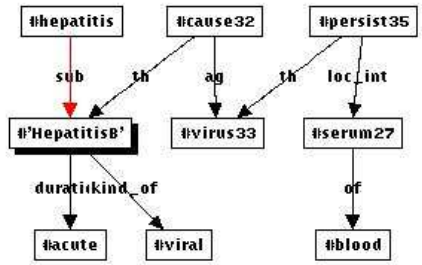


(a3)

Q4: What is caused by a virus that persists in the blood serum?



A4: #'HepatitisB'



(a4)

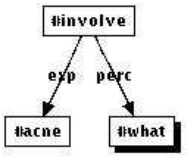
Figure 8.12: OKR for questions and answers with respect to *hepatitis*

we obtain the answer #glands49 (A2 in Figure 8.13). Asking the question *What does acne involve?* (Q1 in Figure 8.13), we obtain the answer #glands55 (A1 in Figure 8.13). Thus, we can obtain the verbalizations corresponding to the particular instances that occur in the definitional text. In this case though, these two instances should be the same, as *oil glands* and *sebaceous glands* are semantically equivalent, but currently we do not consider synonymy. The system could be extended in this direction (see the end of this section for a discussion). Another point that should be mentioned about instances of concepts, is that they can be matched together by Property 5. This is what happened with the instance #serum27 from definitions of #HepatitisA and #HepatitisB in Figure 8.10.

In OKR the concepts are part of the sub hierarchy, while the instances of concepts are not. The instances are connected to their concepts by instance_of relations (not shown in the OKR of the text).

We have seen that having concepts and instances of concepts allows us to directly acquire knowledge and to answer questions relevant to particular verbalizations. Another important aspect that contributes to the adequacy of our representation for acquisition and query is the OKR-equivalences we obtain for different syntactic forms. They are mainly related to verbal constructions. Since we deal with terminology, temporal reasoning is not important, and thus we ignore tense and aspect information. For terminology, however, modals and negation need to be taken into account. For example, negation is essential for differentiating *Hepatitis A* and *Hepatitis B*. Among OKR-equivalence we have: 1) active and passive constructions (e.g., question can be in active voice *What characterizes acne?* (Q4 in Figure 8.13), while the definitional text where the answer is derived from contains

Q1: What does acne involve?

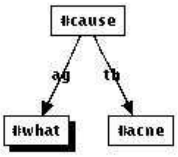


A1: #glands55

```
#involve54.exp= #acne
#involve54.perc= #glands55
#glands55.kind_of= #sebaceous
#glands55.of= #skin
```

(a1)

Q2: What causes acne?

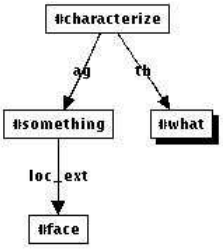


A2: #glands49

```
#cause48.ag= #glands49
#glands49.intensity= #overactive
#glands49.kind_of= #oil
#cause48.th= #acne
```

(a2)

Q3: What is characterized by **something** on the face?

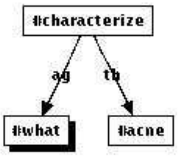


A3: #acne

```
disease.sub= #acne
#characterize41.ag= #pimples42
#appear43.mod=can
#appear43.th= #pimples42
#appear43.loc_ext= #part44
#part44.of= #body
#characterize41.th= #acne
#cause48.ag= #glands49
#glands49.intensity= #overactive
#glands49.kind_of= #oil
#cause48.th= #acne
#involve54.exp= #acne
#involve54.perc= #glands55
#glands55.kind_of= #sebaceous
#glands55.of= #skin
#characterize59.ag=[#papules,or,#pustules,or,#comedones]
#characterize59.th= #acne
#characterize67.ag=[#papules69,and,#pustules70]
[#papules69,and,#pustules70].loc_ext=[#face,and,#neck]
#characterize67.th= #acne
#acne.kind_of= #inflammatory
#acne.body_part= #skin
```

(a3)

Q4: What characterizes acne?



A4-1: #pimples42

```
#characterize41.ag= #pimples42
#appear43.mod=can
#appear43.th= #pimples42
#appear43.loc_ext= #part44
#part44.of= #body
#characterize41.th= #acne
```

A4-2:
[#papules,or,#pustules,or,#comedones]

```
#characterize59.ag=[#papules,or,#pustules,or,#comedones]
#characterize59.th= #acne
```

A4-3: [#papules69,and,#pustules70]

```
#characterize67.ag=[#papules69,and,#pustules70]
[#papules69,and,#pustules70].loc_ext=[#face,and,#neck]
#characterize67.th= #acne
```

(a4)

Figure 8.13: OKR for questions and answers with respect to *acne*

a passive construction *Acne is characterized by papules or pustules or comedones* (A4-2 in Figure 8.13); 2) *-ed* and *-ing* verb forms in reduced relative clauses are equivalent to passive/active verbal constructions (e.g., the question can be formulated in present tense, active voice *What causes hepatitis?* (Q2 in Figure 8.12), while the answer is obtained from a definitional statement involving the reduced relative clause *hepatitis is a disease caused by infectious or toxic agents ...* (A2 in Figure 8.12)); 3) constructions involving raising verbs, where we can take advantage that the controller is not the semantic argument of the raising verb (e.g., in the definition of Hepatitis B we have *... caused by a virus that tends to persist in the blood serum*, while the question can be asked without the raising verb *What is caused by a virus that persists in the blood serum?*).

In all of the above examples, we dealt with precise questions. Our representation, however, facilitates vague questions as well, such as, *What is caused by something that does not persist in the blood serum?* (Q3 in Figure 8.12), or *What is characterized by something on the face?* (Q3 in Figure 8.13). This is obtained by considering *something* as a variable concept that matches a vertex in the OKR. A practical advantage is that we can obtain all the concepts that are in a particular relation with other concepts, or that have particular properties. For example, we can ask the question *What causes something?*. In this case, considering both OKRs in Figure 8.10 and Figure 8.11, *something* would match #acne, #hepatitis, # HepatitisA, and #HepatitisB, showing that they can all be caused by something. The answers are #glands49, #agents7, #virus25, and #virus33, respectively.

In these examples, and in the entire acquisition experiment, the direct knowledge acquisition from text is done only in the context of the weak concept identity

given by Principle 7, which was presented at the beginning of this chapter (i.e., Properties 5 and 6). Principle 7 does not fully guarantee the informal definition of the concept identity principle by which a referent has one and only one vertex in OKR. We call the principle guaranteeing this informal definition, the *strong concept identity principle*. In order to apply the strong concept identity principle in general, Property 5 and 6 should be extended to handle semantic/pragmatic equivalences (e.g., synonyms, anaphora), which is outside the scope of this dissertation. In our terminological knowledge acquisition experiment, only the defined concept(definiendum) is guaranteed to satisfy the strong concept identity principle, while any other referred concepts or instances of concepts that are part of the definiens are guaranteed to satisfy only the weak concept identity given by Principle 7. For example, in Figure 8.13 we can see that the answer concept #glands55 (A1), and the answer concept #glands49 (A2) have the same referent, even if they appear as distinct vertices in OKR. The same holds true for the answers in association with question Q4, which contain concepts or instances of concepts that have the same referent (#pimples42, [#papule,or,#pustules,or,#comedones], [#papules69,and,#pustules70]). These examples show that Property 5 and Property 6 must be refined to handle OKR subDAGs semantic equivalences, which implies the treatment of contradictions as well. While we do not treat these types of semantic equivalences/contradictions in this dissertation, our representation can facilitate the direct acquisition of knowledge that will enable this type of reasoning. Let us consider the two utterances in (6):

(6) a. acne is characterized by pimples.

b. acne is characterized by oil glands.

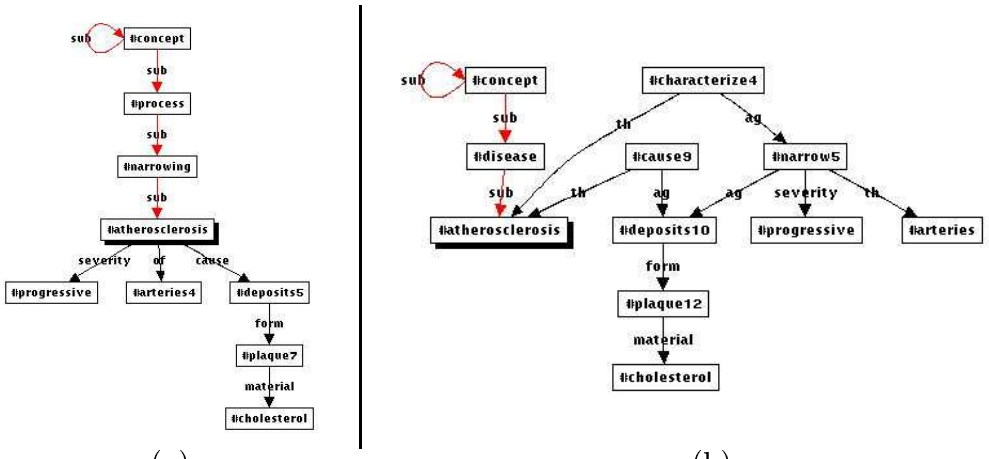


Figure 8.14: OKR for the definition of *atherosclerosis* in Figure 8.8 considering a weak (a), or a strong (b) semantic context interpretation

If we do not have any type of prior knowledge regarding these utterances, they can be added to the ontology. However, suppose we have the knowledge previously acquired from utterances (7) in the ontology:

(7) acne is a disease.

a pimple is a symptom.

a gland is an organ.

a disease is characterized by symptoms.

In this case, using this knowledge allows us to accept the statement in (6a), while rejecting the one in (6b). This type of reasoning is straightforward for any semantic interpreter that uses the generative ontology axioms defined in Section 4.2.

Moreover, a hierarchy of concepts and semantic roles incrementally built, will allow us to replace the weak semantic context used by our semantic interpreter with an incrementally stronger semantic context, when transforming OntoSeR$^-$

into OntoSeR$^+$ (i.e., Φ_{onto}), and TKR into OKR (see Figure 6.2 in Chapter 6). The use of a stronger semantic context will allow us to directly acquire knowledge from increasingly complex and ambiguous utterances.

The difference between having a strong and a weak semantic context becomes crucial when we have semantically complex utterances such as the definition of "atherosclerosis" given in Figure 8.8: *atherosclerosis is the progressive narrowing of arteries from cholesterol plaque deposits.* In a weak semantic context, we have the OKR given in Figure 8.14(a), where *narrowing* is a "process" and is the genus-phrase of the definition. In this example, *narrowing* is a noun, and the copula *to be* is *predicative*. This is a valid interpretation where *atherosclerosis* is the process of narrowing of the arteries.

However, this definition might have a different interpretation if we consider a stronger semantic context, such as the one given by utterances in (8):

(8) atherosclerosis is a disease.
a disease is characterized by symptoms.
a disease is caused by chemical compounds.
the narrowing of the arteries is a symptom.
cholesterol plaque deposits are chemical compounds.

Taking this knowledge into account, the definition of "atherosclerosis" becomes equivalent to the utterances given below:

(9) atherosclerosis is a disease.
atherosclerosis is characterized by the progressive narrowing of the arteries.

atherosclerosis is caused by cholesterol plaque deposits.

arteries are narrowed by cholesterol plaque deposits.

These utterances have the OKR given in Figure 8.14(b), which we can obtain since we have a treatment of nominalizations. But such a representation cannot be obtained from the initial definition alone using only a weak semantic context. However, we do this analysis to see how such an OKR could be obtained directly, given the strong semantic context in (8), and a treatment of the copula *be-equative*. The interpretation of the copula *be-equative* in conjunction with the nominalization and the strong semantic context, should give us that the agent of *narrowing* (i.e., *cholesterol plaque deposits*) is the cause of *atherosclerosis*, as well (see Figure 8.14(b)). This looks similar to the behavior of control verbs, where the argument of the matrix verb becomes also the argument of the embedded verb. Unlike control verbs, where this decision is taken at the lexicon/grammar rule level, the decision is taken in this case at the interpretation level, based on the strong semantic context. In the absence of such a strong context, we cannot have this interpretation. In this dissertation we do not treat such semantically complex definitions, but the goal of this example and of all the examples given in this section is to show that OntoSeR is adequate to directly obtain ontological knowledge from text given a weaker or a stronger semantic context, being a qualitative evaluation of our semantic representation.

8.3 Meaning as Answers to Questions

In Section 4.2 we theoretically defined the meaning of a syntagma, σ, (3.20) and of a discourse, d, (3.21) as the total number of query-syntagmas together with their answers which *become* ground derivable by adding the assertional form K_σ and K_d to the system's knowledge.

In this chapter and Chapter 6, we have introduced the ontological-based representation form for K_σ, K_d, K_o (syntagma, discourse, and ontology, respectively). Every syntagma/discourse is represented at the ontology-level representation, OKR. Moreover, the associated questions and answers are also represented at the OKR level, as we have seen in all the examples presented in these two chapters, where we gave sample questions and their answer(s) related to every example.

In Section 8.2.3 we have seen that the questions can be precise or vague. Thus, it becomes important to define the meaning of a question with respect to a terminological knowledge base, K_o.[7] The meaning of questions and answers has been studied in general by formal theories (Groenendijk and Stokhof, 1984). We give a definition of the meaning of a question and an answer with respect to our framework.

Definition 36. *The meaning of a question, q, with respect to a terminological knowledge base, K_o, is the set of all answers that can be directly obtained from K_o.*

Definition 37. *The answer to a question is the concept that matches the wh-word through the DAG matching algorithm between the question's subDAG and the terminological knowledge base DAG, K_o.*[8]

[7] In this dissertation, we only refer to terminology, where we have clearly defined the concept identity principle.

[8] Definition 37 can be extended if the DAG matching algorithm includes reasoning as well. Complex reasoning can affect tractability. However, in Chapter 4 we have stipulated the principle of natural language as problem formulation (i.e., without reasoning).

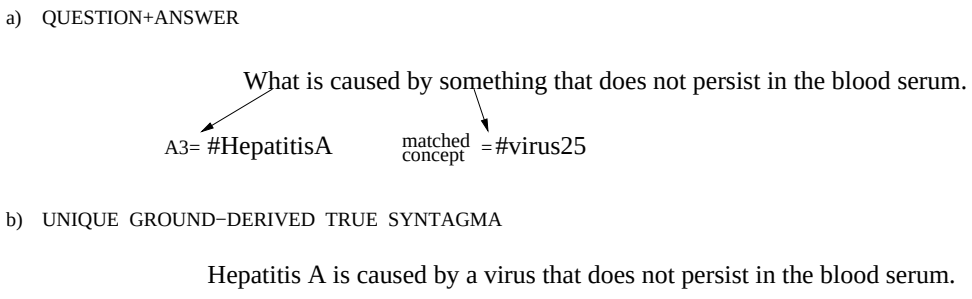


Figure 8.15: Meaning as answers to questions

A question together with an answer determines a unique syntagma that is ground derived from the terminological knowledge base. In Figure 8.15 we present an example to illustrate this fact. The question *What is caused by something that does not persist in the blood serum?* (Q3 from Figure 8.12), together with the tuple consisting of the answer concept A3=#HepatitisA, and the variable concept *something* that matches the concept #virus25, uniquely determines the syntagma corresponding to the utterance *Hepatitis A is caused by a virus that does not persist in the blood serum*, which can be ground derived as true from the knowledge base. In this way, the meaning becomes the set of all syntagmas that can be ground derived from the knowledge base, and whose truth value is true. Thus, our definition of meaning is equivalent to those that use truth conditions, although we use only the concept of ground syntagma derivation based on which we have defined the LWFG semantics (see Chapter 4, (4.20), (4.21), pg. 88). We obtain meaning without using full first-order logic formulas (Blackburn and Bos, 2005).

Unlike meaning as truth conditions, where the problem of meaning equivalence is reduced to logical form equivalence, in our case meaning equivalence is reduced to semantic equivalence of DAGs/subDAGs, which obey the concept iden-

tity principle (weak, or strong). The matching algorithm obtains the same answers to questions, relative to semantic equivalent DAGs. If we consider only the weak concept identity principle given by Properties 5 and 6, the problem is reduced to DAG/subDAG identity.

For example, for terminological knowledge we ignore tense and aspect information as we do not deal with temporal reasoning. In this scenario, the two DAGs (OKRs) corresponding to the utterances in (10) are identical (the subDAG corresponding to the vertex #characterize59 in Figure 8.11), thus their meaning is identical.

(10) a. Acne is characterized by papules or pustules or comedones.

b. Papules or pustules or comedones characterized acne.

As we define meaning through question and answer we need to discuss also the meaning equivalences for questions. Let us consider the two vague questions in (11):

(11) a. What characterizes something?

b. What was characterized by something?

Their corresponding OKRs are given in Figure 8.16. Let us call *what* the "answer concept" and *something* the "matched concept". These two OKRs become identical if the agent ag and the theme th slots for the concept #characterize are the same concepts in the two OKRs. That is the pair ("answer concept", "matched concept") of the first question, is identical with the pair ("matched concept", "answer concept") of the second question. As we saw in Figure 8.15 a question together with an

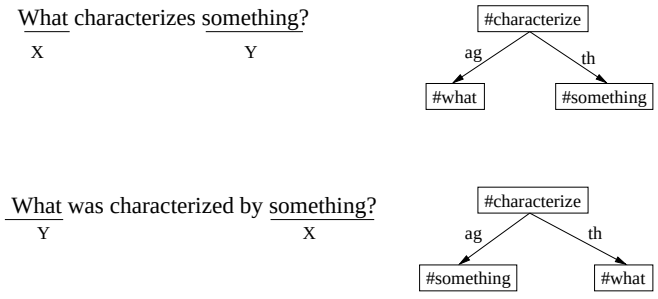


Figure 8.16: Semantic equivalence of questions

answer determines a unique syntagma that can be ground derived as true. Given this, the vague question equivalence becomes the equivalence of "X characterizes Y", "Y was characterized by X", which we showed through example (10) that are equivalent (we ignore tense).

The above discussion motivates our approach of considering meaning as answers to questions, defined in Section 4.2, where the meaning of a knowledge base K_o, is given by all syntagmas that result from question-answer pairs, and which can be ground derived as true from the knowledge base K_o (see (3.20)). Chapter 6 and Chapter 8, which constitute a *qualitative evaluation*, show the adequacy of OntoSeR to enable the practical use of the theoretical definition of meaning given in Section 4.2, in the context of terminological knowledge, where we have the concept identity principle well defined. From an applicative perspective, the evaluation of OntoSeR can be performed by any user by providing a set of questions with respect to a given text and evaluating the obtained concept-level answers. If correct answers are obtained for all possible questions, it means that the representation is correct. We use this type of evaluation in the pilot experiment presented in the next section.

8.4 Acquisition and Querying of OKR-annotated treebanks — Pilot Experiment

The goal of this section is to show that our semantic representation, the learned grammar, and our semantic interpreter can be used to build semi-automatically OKR-annotated treebanks suitable for NL-querying, where precise/vague questions can be asked and precise answers (at the concept level) are always obtained.

This setting is different from the question-answering tasks of the Text REtrieval Conference (TREC) (Voorhees, 1999) and the Advanced Question and Answering for Intelligence (AQUAINT) program, which involve finding answers in large collections of documents, with or without reasoning. In our case the text consists of one or more utterances, and we are interested in factual answers that are explicitly stated in those utterances (i.e., no reasoning is involved). The reader should keep in mind that the acquisition/querying setting is used to evaluate qualitatively our LWFG learning model (the learned grammar and our semantic representation). Our setting is also different from Natural Language Interfaces to Databases (NLDIB) since we automatically acquire the knowledge base from text. After the acquisition takes place, we can see the question as a NL-query to a knowledge base.

In order to carry out this qualitative evaluation, we have used small sets of utterances of three types: 1) 73 examples used to show individual linguistic phenomena, which we selected from literature and Quirk grammar (Quirk et al., 1972); 2) 15 complex utterances that combined difficult linguistic phenomena which we selected from news-like articles; 3) set of 17 complex definitions in the medical domain automatically extracted by DEFINDER from on-line medical articles.

Samples from the first two sets have been used in the examples in Chapter

6. The last set has been exemplified in this chapter and has been used to report the results of building a pilot OKR-annotated treebank and performing a querying experiment. Before describing this pilot experiment, we discuss how we acquired the lexicon and the ontological information for the weak semantic context used by our semantic interpreter.

For all the data sets we have derived a lexicon from COMLEX (Grishman, Macleod, and Meyers, 1994) and UMLS lexicon (Lindberg, Humphreys, and McCray, 1993), which contains information that is needed for the elementary semantic molecules. For our weak semantic context we needed information regarding the semantic roles of verbs, prepositions, attributes of adjectives, adverbs and also nouns that appear in noun-noun compounds. For the semantic roles of verbs and prepositions we extracted the thematic roles from the "LCS Database" (Dorr, 1997). For adjectives and adverbs we used information from WordNet (Miller, 1990). However, especially for the medical definitions, these resources do not contain all the required information and thus we were forced to manually introduce this information (especially for adjectives, nouns, and specific roles of prepositions).

Acquisition of a pilot OKR-annotated treebank. In this experiment we tested the use of our learned grammar and of our semantic interpreter based on weak semantic context to build a pilot OKR-annotated treebank for terminological knowledge. We used the set of 17 medical definitions. In Appendix C.1 we show these definitions as well as the number of different syntagmas obtained without and with the semantic validation (Φ_{onto}). Without semantic validation, the average number of syntagmas (OntoSeR$^-$) obtained by our robust parser is 2.53 per definition. After Φ_{onto} is applied, the average number of different syntagmas (OntoSeR$^+$)

obtained for a definition is 2.00. We noticed that the weak semantic context that our semantic interpreter implements is not enough to obtain only the correct semantic analysis. Thus, we developed the system to allow a user to manually select the correct OKR and to add it to a treebank. The selection of the OKR-level of representation for human validation is due to the fact that this representation is much more "readable" for a user than the OntoSeR$^-$ and OntoSeR$^+$ levels, as can be seen from Appendix B. This mode of operation allows the semi-automatic creation of OKR-annotated treebanks, with user validation. Building such a treebank is important for further developing the semantic interpreter towards strong semantic context. In Appendix C.1 we give the 17 medical definitions and in Appendix C.2 their OKR-annotated treebank.

NL-querying of the acquired treebank. For this experiment, we created a benchmark of 29 questions. We used both precise (22) and vague (7) questions. The type of questions we used are "Who did what to whom?", that is only questions regarding the verbs' arguments. Since in our OKR treebank we obtained a hierarchy of concepts, the question can be related to this hierarchy: e.g., the question *Which are viral diseases?* has as answer #HepatitisA and #HepatitisB, even if their direct parent is #hepatitis and not #disease. For questions, the ambiguity is increased due to our treatment of long distance dependencies. As can be seen from Appendix C.3, for questions we have an average of 6.06 syntagmas per question at the OntoSeR$^-$ level (i.e., without Φ_{onto} validation). This is explained by our treatment of long distance dependencies. After semantic validation, we have an average of 2.35 syntagmas per question. In this experiment though, even if the weak semantic context is not always enough to eliminate incorrect semantic representations of

questions, we only obtain the correct answer(s), since we match the OKRs of these questions against the manually validated correct treebank. In Appendix C.3 we give all the questions, with the number of syntagmas at the OntoSeR$^-$ level, at the OntoSeR$^+$ level and with the answer concepts. We see that we obtain precise answers at the concept level. We also give a sample of these question-answer(s) pairs that contains the answers together with their subDAGs (Appendix C.4).

Appendix C presents all the data of this experiment. We conclude this chapter with a discussion of the outcome of our qualitative evaluation:

Positive Conclusions. OntoSeR allows us to acquire ontological knowledge in OKR form that obeys the *weak concept identity principle*. As we deal only with a weak semantic context given by the admissibility relations that we can find at the level of lexical entries, our qualitative evaluations show that "a lexicon can sometimes be the basis for the development of a practical ontology" (Hirst, 2003). The weak concept identity guarantees the same OKR representation for different grammatical forms (e.g., nominalizations - verbal forms; active-passive voice; *-ing* and *-ed* forms of reduced relative clauses - active/passive forms of verbs), or different forms of tense and aspect, which are filtered. Since we focus on terminological knowledge, modals and negation are important, while temporal reasoning is not. However, if we would not filter the tense and aspect, the semantic interpreter could be further developed towards this reasoning needed for factual knowledge bases. Another important conclusion of our evaluation is that the meaning seen as question together with its answer allows the ground derivation of syntagmas even from the filtered form of OKR (for which we do not have the reversibility property anymore).

Open Problems. The main current limitation of our semantic interpreter is that it

uses only a weak semantic context. We do not currently use hierarchies of concepts and semantic roles, and semantic equivalences based on synonymy and anaphora. However, we mentioned in this chapter that our framework can be used in future work to incrementally enhance the weak semantic context towards a strong semantic context by acquiring this knowledge directly from natural language utterances. The learned experimental grammar will require further revision that goes up to the refinement of some elementary semantic molecules: adverb, relative pronoun (in order to be able to treat *whose*, *where*, *when*, *how*, that will require a different semantic molecule), and the coordinator *or*. However, the main open problem that was noticed is the ambiguity which appears in the examples with complex coordination constructions characteristic of the medical corpus. This problem emphasizes the need to have an interaction between the constraints Φ_{onto} and Φ_{comp}, by introducing a semantic attribute in the head of semantic molecules, whose value should be provided by the Φ_{onto} constraint during parsing. For example, the utterance *comedones, papules and pustules on the face, neck and upper body* is highly ambiguous (the coordination rule is both right and left recursive). The presence of a semantic attribute (e.g., symptom for *comedones*, *papules* and *pustules*, and body_part for *face*, *neck*, and *upper body*) in the head of the semantic molecules, and thus at the level of the compositional constraint Φ_{comp}, allows only two groupings. In our experiment we simulated such an effect by using the attribute count — which is available at the lexical level and is present in the head of semantic molecules of nouns — to discriminate between the two groups of nouns. In this way we noticed the decrease in ambiguity, which in the future should be captured by the interaction between Φ_{onto} and Φ_{comp}.

The pilot experiment has shown that we can build semi-automatically OKR-annoted treebanks, from definitions given in natural language text. This can be used in the future to enhance the OKR with probabilities. This probabilistic enhancement only at the conceptual level bears similarity to Pinker's theory of child language acquisition that there is "a formal and nearly exceptionless grammatical linkage between syntax and semantics, and a more probabilistic cognitive correlation between semantics in parental speech and childlike concepts" (Pinker, 1989, page 364). Our current system requires manual validation of the obtained OKR (since the level of ambiguity is still fairly high). Adding probabilities at the OKR level will eliminate the need for manual validation, and thus will allow a future quantitative evaluation of our grammar learning model based on question-answering experiments on unannotated text. The corpus used for LWFG learning evaluation should contain precise answers explicitly stated in the text (i.e., without reasoning).

Chapter 9

Conclusions

We conclude this dissertation with a summary of the main contributions of this work. We also discuss the open problems and limitations and propose directions for future work.

9.1 Contributions

The fundamental idea which was the starting point of this dissertation was the introduction of a new semantic representation, OntoSeR which should allow deep language understanding and should have the properties of an ontology query language. The practical use of this representation would have required us to rewrite existing grammars by hand in order to facilitate access to ontology-based meaning during parsing. As a consequence, we posed the problem of defining a new type of grammar suitable for deep language understanding, which is learnable using relational learning methods, eliminating thus the need of rewriting hand-crafted grammars.

This dissertation is the result of addressing this problem, whereby the defined theoretical concepts are original. Following is a summary of both the theoretical

and application-oriented contributions of this work. While the research in this dissertation spans several topics, the common thread we follow is the learnability issue.

9.1.1 Theoretical Contributions

New Representation of Natural Language. We have defined three levels of representation in this dissertation:

- **Semantic Molecule — representation at the grammar level.** We introduced a new syntactico-semantic representation of natural language expressions, which we call *semantic molecule*. The semantic molecule of a natural language string w has two components — the head and the body — $w' = \binom{h}{b}$. The head is a flat feature structure which contains syntactico-semantic attributes needed for composition. The body of the semantic molecule is a new flat semantic representation, OntoSeR. This representation is the concatenation of the semantic representations of lexical items it contains, followed by a variable substitution obtained as a consequence of applying the semantic composition constraints Φ_{comp} during parsing. The novelty of OntoSeR is twofold: it is an ontology query language, and it ensures the reversibility of parsing/generation, without the information from the head of the semantic molecule. Our semantic molecule representation allowed us to define the semantic composition and semantic interpretation as grammar constraints at the rule level. The semantic composition is given by constraints applied to the heads of the semantic molecules, while the semantic interpretation is given by an ontology constraint applied to the body of the semantic molecule. The

compositional constraints are learned together with the grammar rules, while the ontology constraint provides access to meaning during parsing. We have formulated the properties and principles of the semantic molecule representation.

- **Text Knowledge Representation (TKR) — representation at the text level.** TKR is an asserted form of OntoSeR, which represents the entire text. We still have at this level the reversibility property (i.e., from this representation we can directly obtain the text through our parser/generator).

- **Ontology-level Knowledge Representation (OKR) — representation at the ontology level.** From TKR we can obtain the OKR representation through filtering and by introducing the concept identity principle, which establishes a bijection between a concept and a referent. We define the OKR as a directed acyclic graph (DAG), where vertices are concepts or instances of concepts, and edges are semantic roles (including properties). In this dissertation we only define and implement a weak concept identity principle.

Lexicalized Well-Founded Grammars — A New Type of Constraint-based Grammars. In this dissertation we introduced a learnable type of Definite Clause Grammars, which we call Lexicalized Well-Founded Grammars (LWFGs). These grammars capture both syntax and semantics and have constraints at the rule level for semantic composition and semantic interpretation. The nonterminals of a LWFG are augmented with syntagmas, which are pairs of strings and their semantic molecules ($\sigma = (w, w')$). Moreover, LWFGs introduce a partial ordering among nonterminals, which allows the ordering of syntagmas and thus the bottom-

up induction of these grammars. The learning is done from a set of representative examples, which generates the grammar rules, and a representative sublanguage, which gives the reduced semantics used for generalization. We have formally defined the representative examples and the representative sublanguage, and provided an efficient algorithm that generates the representative example set, given a grammar and a representative sublanguage. We also defined the semantics of LWFGs and the robust parsing/generation for LWFGs as deduction. We have proved that our Lexicalized Well-Founded Grammars are decidable.

Grammar Approximation by Representative Sublanguage — A New Model for Relational Learning of LWFGs from Positive-only Examples. In this dissertation, we have defined the LWFG learning as a decidable Inductive Logic Programming problem. The decidability of ILP is a consequence of LWFG decidability, which is guaranteed by the fact that we chose as provability relation of ILP, $\vdash$, the robust parsing provability, $\vdash_{rp}$. We have defined the *representative example parsing preserving property* of LWFGs, which allowed us to prove that the search space for grammar learning is a complete grammar lattice/grammar boolean algebra. Based on this search space property, our learning method can generate the most specific hypotheses from the representative examples, and then generalize these hypotheses based on the reduced semantics defined by the representative sublanguage. Thus, the learning method is based on inverse entailment, using the reduced semantics as performance criteria. For a type of LWFGs (conformal) we proved the *learnability theorem*, which states that a conformal Lexicalized Well-Founded Grammar can always be learned from a representative example set and a conformal representative sublanguage, the solution being the top element of the

grammar lattice search space. The learnability theorem extends significantly the class of problems learnable by ILP methods. This class is a class of constraint-based grammars which capture syntax and semantics (LWFG) and which are learnable by relational learning methods. We have provided polynomial algorithms for grammar induction based either on ordered representative examples — the search space is a grammar boolean algebra — or on unordered representative examples — the search space is a complete grammar lattice. We proved the soundness of these algorithms. The convergence property of the iterative algorithm from unordered examples allows the sound and straightforward revision of LWFGs. This revision property makes LWFGs suitable for incremental coverage of natural language fragments.

9.1.2 Application-oriented Contributions

Framework for Grammar Learning and Merging. In this dissertation, we implemented a system that represents an experimental platform for all the theoretical algorithms. This system consists of:

- an Inductive Logic Programming system that learns only from positive examples, uses background knowledge and has a dual mode of operation: learns from both ordered and unordered representative examples.

- a robust parser/generator that has been used in all our learning experiments to signal the over/under generalization and ambiguity.

We have learned an experimental grammar of 151 grammar rules and 151 grammar constraints that covers diverse and complex linguistic phenomena, such as raising and control, long-distance dependencies, relative clauses, noun compounds,

nominalizations, coordinations, verbal constructions with auxiliaries. In order to learn this grammar, we have built a semantically annotated treebank of 151 representative examples and 448 representative sublanguage examples. This small annotated treebank represents a new type of annotated corpus used for grammar relational learning. This framework has the practical advantage of implementing sound grammar revision and grammar merging, which allow an incremental coverage of natural language fragments.

Semantic Interpreter For Text-to-Knowledge Acquisition. We built a semantic interpreter based on a weak semantic context. Using this interpreter, we provided qualitative evaluations that show that our semantic representation OntoSeR, and its derived forms TKR and OKR are expressive enough to represent all the above mentioned linguistic phenomena covered by our learned experimental grammar, are useful for direct knowledge acquisition and deep language understanding (concept level answers to questions). The semantic interpreter implements the weak concept identity principle for terminological knowledge. We have shown that the interpreter allows us to semi-automatically build an OKR-annotated treebank (with human validation) from a set of medical definitions, and to perform NL-querying experiments. The implementation of the weak concept identity principle allows us to experiment with the meaning as pairs of questions and their answers. We have shown that for a terminological knowledge base in OKR form, which does not allow the direct generation of text from the representation, the meaning can still be defined as all syntagmas that can be derived as true from the OKR by using pairs of questions and answers. In this way, meaning can be obtained (theoretically and practically) without explicitly using first-order logic.

9.2 Open Problems and Future Work

In this dissertation we proved theoretically and showed practically that Lexicalized Well-Founded Grammars can be learned from a set of representative examples and a representative sublanguage. As a consequence, the following statement holds true:

> If natural language can be covered by Lexicalized Well-Founded Grammars, and the set of representative examples and the representative sublanguage are given, then natural language can be learned.

While in this dissertation we have shown that the representative example and representative sublanguage sets can be built and that diverse and complex fragments of natural language can be covered by Lexicalized Well-Founded Grammars, the **main open problem** that still remains is

> Can natural language, in general, be covered by LWFGs?

The answer to this question is not an immediate one, and only future research might be able to provide a definite answer. However, in this dissertation we provided a learning framework that incrementally learns, thus the system can be used in future research towards broad coverage of language. In order to move towards broad coverage, we must address first the two main limitations of the current work:

9.2.1 Weak Semantic Context/Weak Concept Identity

In this dissertation, we only use a weak semantic context for role/property assignment and a weak concept identity principle. The experimental results have shown that in this situation we have still a fairly high degree of ambiguity. Regarding the property of concept identity — i.e., the bijection between the OKR vertices

and referents — using only the weak concept identity principle we cannot currently treat semantic equivalences, such as synonymy and anaphora.

Future work must consider the incremental enhancement of the semantic context to construct hierarchies of concepts and semantic roles and to add selectional restrictions. We should introduce a strong concept identity principle, which incorporate synonymy and anaphora. The enhancement of the semantic context and the strong concept identity principle open the problem of defining a normal form for the OKR representation. As tractability might be affected by the enhancement of the semantic context, future work should address the issue of introducing probabilities in order to choose the semantic type of a word and to quantify the admissibility of concept association. OKR-annotated treebank(s), which our pilot study showed can be built with the current weak semantic context and human validation, can be used in future work to enrich the hierarchy of concepts with probabilities. The medical definition corpus automatically extracted by DEFINDER from medical articles will be further exploited in this direction.

9.2.2 Evaluation

The evaluation of a grammar learning framework for deep semantic analysis is a difficult problem since we do not have large semantically annotated treebanks and they are difficult to build. Even if our system learns from a reduced-size treebank, in order to evaluate it for broad coverage we would need a large annotated treebank. The lack of a large OntoSeR-annotated treebank has led us to evaluate our framework in two ways: 1) a qualitative evaluation regarding the coverage of diverse and difficult linguistic phenomena, and 2) a pilot experiment of acquisition and NL-querying of terminological knowledge from unannotated text (i.e., definitions).

For the latter we only used a small set since it requires human validation of the pilot OKR-annotated treebank.

In future work, as the semantic interpreter is enhanced with a strong semantic context with probabilities, such a system can be used to evaluate our grammar learning framework and our representation on a larger scale using just unannotated text. The main characteristic of the corpora used for this evaluation purpose, is that the questions should refer only to what was stated explicitly in text. Thus, they should target the principle of natural language as problem formulation (i.e., without deep reasoning). Building such a corpus of question and answers is less difficult than building a large semantically annotated treebank, and the evaluation can be performed by any type of users, as the answers that our system gives are precise, at the concept level.

As our grammar becomes one of broad coverage, another **open problem** would be whether our semantic representation OntoSeR can be used for merging existing treebanks that focus on parts of the larger problem of semantic annotation. Pustejovsky et al. (2005) discuss the issues involved in merging four of these efforts into a unified linguistic structure: PropBank (Palmer, Gildea, and Kingsbury, 2005), NomBank (Meyers et al., 2004), the Discourse Treebank (Miltsakaki et al., 2004) and Coreference Annotation (Poesio and Vieira, 1998). The issues are to resolve overlapping and conflicting annotations as well as to investigate how the various annotation schemes can reinforce each other to produce a representation that is greater than the sum of its parts. It would be an interesting future direction to investigate how OntoSeR and its derived representation forms TKR and OKR could be used to realize such a merging automatically.

Appendix A

Learning an Experimental Grammar

This appendix presents run examples that illustrate our grammar learning framework. In Appendix A.1 we present the concepts of learning from ordered and unordered representative examples, for the case of auxiliary verbs. Appendix A.2 shows an example of one iteration of the grammar learning algorithm (i.e., generation of the most specific rule, the generalization process and the best rule chosen based on performance criteria). In Appendix A.3 we give several examples of parsing and generation, which exemplify the reversibility of our grammar. We give examples of learned grammar rules and constraints for several fragments of natural language in Appendix A.4. Appendix A.5 shows through an example our treatment of long-distance dependencies. We conclude this appendix with an illustration of how to generate representative examples given a grammar and a sublanguage using Algorithm 4 presented in Section 3.6 (Appendix A.6).

The runs are given in Prolog. The grammar rules are shown in their DCG form, where the constraints are enclosed in {}. The syntagmas are represented as follows:

```
([skin,disease],[cat:na,nr:sg,head:Y,mod:Z]><[X@is_a:skin,Y@P:X,
 Y@is_a:disease,Z@P1:Y])
```

where the first element is the string given as a Prolog list, while the second element `h><b` is the corresponding semantic molecule (`h` is the head of the molecule, `b` is the body, and both are given as Prolog lists). The atomic predicates of the body `b` are represented as `concept@attr:concept`, which is just a Prolog notational variant for the form *concept.attr=concept*, which we used throughout this dissertation. Capital letters denote variables.

A.1 Learning Auxiliary Verbs

In this section we present the experiment of learning a grammar for finite auxiliary verbs. We show the representative examples used, the learned grammar rules and samples of the learned compositional constraints. We experimented with both ordered representative examples (Appendix A.1.1) and unordered representative examples (Appendix A.1.2) as input. As can be noticed, the learned grammar is the same in both cases.

A.1.1 Learning From Ordered Representative Examples

In the first experiment we used our algorithm *Constraint_Grammar_Induction*(E_R, E_σ, K) that learns from an *ordered* set of representative examples (see Algorithm 7, Section 5.1.2). The ordered representative example set, E_R, the learned grammar G and samples of the learned compositional constraints are given below. As can be seen the number of representative examples equals the number of learned rules, in this case 14. A representative sublanguage E_σ consisting of 81 examples was used for rule generalization.

Ordered representative examples, E_R given as input:[1]

```
======================================================================
pos([he],[cat:sbj,stype:s,dets: y,pers:(_,3),nr:sg,case:((n,ng)),hum:y,
    head:X]><[X@is_a:he]).
pos([john],[cat:sbj,stype:s,dets:y,pers:(_,3),nr:sg,case:((n,ng)),hum:y,
    head:X]><[X@name:john]).
pos([who],[cat:sbj,stype:s,dets:no,pers:(_,3),nr:_,case:((n,ng)),hum:y,
    head:X]><[X@is_a:who]).
pos([someone,is],[cat:av0,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),
    nr:sg,pf:no,pg:no,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr]).
pos([is,someone],[cat:av0,stype:s,vtype:aux,vft:fin,int:y,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[Y@tense:pr, X@is_a:someone]).
pos([who,is],[cat:av0,stype:s,vtype:aux,vft:fin,int:_,dets:no,
    case:((n,ng)),hum:y,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:who, Y@tense:pr]).
pos([someone,is],[cat:av1,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr]).
pos([someone,is,not],[cat:av1,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:y,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone,Y@tense:pr,Y@neg:y]).
pos([someone,is],[cat:av2,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr]).
pos([someone,can,be],[cat:av2,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:mod,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone,Y@mod:can,Y@bse:be]).
pos([someone,is],[cat:av3,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr]).
pos([she,has,been],[cat:av3,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:y,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:y,
    pg:no,headS:X,head:Y]><[X@is_a:she,Y@tense:pr,Y@pf:y]).
pos([someone,is],[cat:av4,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr]).
pos([someone,is,being],[cat:av4,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:y,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr,Y@pg:y]).
======================================================================
```

[1]It can be noticed that our input for grammar learning does not consist of entire sentences that are annotated as is customary when learning from the Penn Treebank. Our examples are syntagmas (which can be any unit corresponding to words, phrases, clauses and sentences). Thus our representative examples are a different type of annotated corpus for grammar learning.

The learned LWFG rules in DCG form:

```
=====================================================================
sbj(A><B-C)--->pro(D><B-C),{phi_comp(1,[A,D]),phi_onto(B)}.
sbj(A><B-C)--->pn(D><B-C),{phi_comp(2,[A,D]),phi_onto(B)}.
sbj(A><B-C)--->relpro(D><B-C),{phi_comp(3,[A,D]),phi_onto(B)}.
av0(A><B-C)--->sbj(D><B-E),aux(F><E-C),{phi_comp(4,[A,D,F]),phi_onto(B)}.
av0(A><B-C)--->aux(D><B-E),sbj(F><E-C),{phi_comp(5,[A,D,F]),phi_onto(B)}.
av0(A><B-C)--->relpro(D><B-E),aux(F><E-C),{phi_comp(6,[A,D,F]),phi_onto(B)}.
av1(A><B-C)--->av0(D><B-C),{phi_comp(7,[A,D]),phi_onto(B)}.
av1(A><B-C)--->av0(D><B-E),aux(F><E-C),{phi_comp(8,[A,D,F]),phi_onto(B)}.
av2(A><B-C)--->av1(D><B-C),{phi_comp(9,[A,D]),phi_onto(B)}.
av2(A><B-C)--->av1(D><B-E),aux(F><E-C),{phi_comp(10,[A,D,F]),phi_onto(B)}.
av3(A><B-C)--->av2(D><B-C),{phi_comp(11,[A,D]),phi_onto(B)}.
av3(A><B-C)--->av2(D><B-E),aux(F><E-C),{phi_comp(12,[A,D,F]),phi_onto(B)}.
av4(A><B-C)--->av3(D><B-C),{phi_comp(13,[A,D]),phi_onto(B)}.
av4(A><B-C)--->av3(D><B-E),aux(F><E-C),{phi_comp(14,[A,D,F]),phi_onto(B)}.
=====================================================================
```

Samples of learned compositional constraints predicates (Φ_{comp}):[2]

```
=====================================================================
phi_comp(3, [A,C]) :-
        eq(A, head:D, C, head:D),
        eq(A, cat:sbj),
        eq(A, stype:s),
        eq(A, dets:no, C, dets:no),
        eq(A, pers:(E,F), C, pers:(E,F)),
        eq(A, nr:G, C, nr:G),
        eq(A, case:(n,ng), C, case:(n,ng)),
        eq(A, hum:H, C, hum:H),
        eq(C, cat:relpro),
        e_list(A).
phi_comp(4, [A,C,D]) :-
        eq(A, headS:E, C, head:E),
        eq(A, head:F, D, head:F),
        eq(A, cat:av0),
        eq(A, stype:G, C, stype:G),
        eq(A, vtype:H, D, vtype:H),
        eq(A, vft:fin, D, vft:fin),
        eq(A, int:no),
        eq(A, dets:y, C, dets:y),
        eq(A, case:(I,J), C, case:(I,J)),
        eq(A, hum:K, C, hum:K),
        eq(A, aux:L, D, aux:L),
        eq(A, neg:M, D, neg:M),
```

[2]The first argument of the `phi_comp` predicates is the number of the grammar rule that these constraints are associated with.

```
        eq(A, tense:N, D, tense:N),
        eq(A, pers:(O,P), C, pers:(O,P)),
        eq(A, pers:(Q,R), D, pers:(Q,R)),
        eq(A, nr:S, C, nr:S),
        eq(A, nr:T, D, nr:T),
        eq(A, pf:U, D, pf:U),
        eq(A, pg:V, D, pg:V),
        eq(C, cat:sbj),
        eq(D, cat:aux),
        e_list(A).
phi_comp(5, [A,C,D]) :-
        eq(A, headS:E, D, head:E),
        eq(A, head:F, C, head:F),
        eq(A, cat:av0),
        eq(A, stype:G, D, stype:G),
        eq(A, vtype:H, C, vtype:H),
        eq(A, vft:fin, C, vft:fin),
        eq(A, int:y),
        eq(A, dets:y, D, dets:y),
        eq(A, case:(I,J), D, case:(I,J)),
        eq(A, hum:K, D, hum:K),
        eq(A, aux:L, C, aux:L),
        eq(A, neg:M, C, neg:M),
        eq(A, tense:N, C, tense:N),
        eq(A, pers:(O,P), C, pers:(O,P)),
        eq(A, pers:(Q,R), D, pers:(Q,R)),
        eq(A, nr:S, C, nr:S),
        eq(A, nr:T, D, nr:T),
        eq(A, pf:U, C, pf:U),
        eq(A, pg:V, C, pg:V),
        eq(C, cat:aux),
        eq(D, cat:sbj),
        e_list(A).
phi_comp(7, [A,C]) :-
        eq(A, headS:D, C, headS:D),
        eq(A, head:E, C, head:E),
        eq(A, cat:av1),
        eq(A, stype:F, C, stype:F),
        eq(A, vtype:G, C, vtype:G),
        eq(A, vft:H, C, vft:H),
        eq(A, int:I, C, int:I),
        eq(A, dets:J, C, dets:J),
        eq(A, case:(K,L), C, case:(K,L)),
        eq(A, hum:M, C, hum:M),
        eq(A, aux:N, C, aux:N),
        eq(A, neg:O, C, neg:O),
        eq(A, tense:P, C, tense:P),
        eq(A, pers:(Q,R), C, pers:(Q,R)),
        eq(A, nr:S, C, nr:S),
```

```
        eq(A, pf:T, C, pf:T),
        eq(A, pg:U, C, pg:U),
        eq(C, cat:av0),
        e_list(A).
phi_comp(8, [A,C,D]) :-
        eq(A, headS:E, C, headS:E),
        eq(A, head:F, C, head:F),
        eq(A, head:G, D, head:G),
        eq(A, cat:av1),
        eq(A, stype:H, C, stype:H),
        eq(A, vtype:I, C, vtype:I),
        eq(A, vtype:J, D, vtype:J),
        eq(A, vft:fin, C, vft:fin),
        eq(A, int:K, C, int:K),
        eq(A, dets:L, C, dets:L),
        eq(A, case:(M,N), C, case:(M,N)),
        eq(A, hum:O, C, hum:O),
        eq(A, aux:P, C, aux:P),
        eq(A, neg:y, D, neg:y),
        eq(A, tense:Q, C, tense:Q),
        eq(A, pers:(R,S), C, pers:(R,S)),
        eq(A, nr:T, C, nr:T),
        eq(A, pf:U, C, pf:U),
        eq(A, pg:V, C, pg:V),
        eq(C, cat:av0),
        eq(C, neg:no),
        eq(D, cat:aux),
        eq(D, aux:not),
        e_list(A).
phi_comp(10, [A,C,D]) :-
        eq(A, headS:E, C, headS:E),
        eq(A, head:F, C, head:F),
        eq(A, head:G, D, head:G),
        eq(A, cat:av2),
        eq(A, stype:H, C, stype:H),
        eq(A, vtype:I, C, vtype:I),
        eq(A, vtype:J, D, vtype:J),
        eq(A, vft:fin, C, vft:fin),
        eq(A, int:K, C, int:K),
        eq(A, dets:L, C, dets:L),
        eq(A, case:(M,N), C, case:(M,N)),
        eq(A, hum:O, C, hum:O),
        eq(A, aux:P, D, aux:P),
        eq(A, neg:Q, C, neg:Q),
        eq(A, tense:mod, C, tense:mod),
        eq(A, pers:(R,S), C, pers:(R,S)),
        eq(A, nr:T, C, nr:T),
        eq(A, pf:U, C, pf:U),
        eq(A, pg:V, C, pg:V),
```

```
        eq(C, cat:av1),
        eq(C, aux:mod),
        eq(D, cat:aux),
        eq(D, vft:nf),
        e_list(A).
phi_comp(14, [A,C,D]) :-
        eq(A, headS:E, C, headS:E),
        eq(A, head:F, C, head:F),
        eq(A, head:G, D, head:G),
        eq(A, cat:av4),
        eq(A, stype:H, C, stype:H),
        eq(A, vtype:I, C, vtype:I),
        eq(A, vtype:J, D, vtype:J),
        eq(A, vft:fin, C, vft:fin),
        eq(A, int:K, C, int:K),
        eq(A, dets:L, C, dets:L),
        eq(A, case:(M,N), C, case:(M,N)),
        eq(A, hum:O, C, hum:O),
        eq(A, aux:be, C, aux:be),
        eq(A, aux:be, D, aux:be),
        eq(A, neg:P, C, neg:P),
        eq(A, tense:Q, C, tense:Q),
        eq(A, pers:(R,S), C, pers:(R,S)),
        eq(A, nr:T, C, nr:T),
        eq(A, pf:U, C, pf:U),
        eq(A, pg:y, D, pg:y),
        eq(C, cat:av3),
        eq(C, pg:no),
        eq(D, cat:aux),
        eq(D, vft:nf),
        e_list(A).
=========================================================================
```

A.1.2 Learning from Unordered Representative Examples

In the second experiment of learning a grammar of finite auxiliary verbs, we used our iterative learning algorithm *Iterative_Grammar_Induction*(E_R^u, E_σ, K), which learns from unordered representative examples, E_R^u (see Algorithm 9, Section 5.1.3). We took the set representative examples E_R given in the previous section and scrambled them, obtaining the representative example set E_R^u given below. We show the iteration steps, and the reduced semantics at each step (i.e., the numbered

of covered positive examples from the representative sublanguage, E_σ, used for generalization). At each iteration step we show the best set of grammar rules. We can notice that the grammar obtained at the last iteration is the same as the grammar learned from the ordered representative examples.

Unordered representative examples, E_R^u given as input:

```
======================================================================
pos([he],[cat:sbj,stype:s,dets:y,pers:(_,3),nr:sg,case:((n,ng)),hum:y,
    head:X]><[X@is_a:he]).
pos([john],[cat:sbj,stype:s,dets:y,pers:(_,3),nr:sg,case:((n,ng)),hum:y,
    head:X]><[X@name:john]).
pos([who],[cat:sbj,stype:s,dets:no,pers:(_,3),nr:_,case:((n,ng)),hum:y,
    head:X]><[X@is_a:who]).
pos([someone,is],[cat:av0,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr]).
pos([is,someone],[cat:av0,stype:s,vtype:aux,vft:fin,int:y,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[Y@tense:pr, X@is_a:someone]).
pos([who,is],[cat:av0,stype:s,vtype:aux,vft:fin,int:_,dets:no,
    case:((n,ng)),hum:y,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:who, Y@tense:pr]).
pos([someone,is],[cat:av4,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr]).
pos([someone,is,being],[cat:av4,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:y,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr,Y@pg:y]).
pos([someone,is],[cat:av3,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr]).
pos([she,has,been],[cat:av3,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:y,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:y,
    pg:no,headS:X,head:Y]><[X@is_a:she,Y@tense:pr,Y@pf:y]).
pos([someone,is],[cat:av2,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr]).
pos([someone,can,be],[cat:av2,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:mod,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone,Y@mod:can,Y@bse:be]).
pos([someone,is],[cat:av1,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:no,tense:pr,pers:(_,3),nr:sg,pf:no,
    pg:no,headS:X,head:Y]><[X@is_a:someone, Y@tense:pr]).
pos([someone,is,not],[cat:av1,stype:s,vtype:aux,vft:fin,int:no,dets:y,
    case:(n,ng),hum:_,aux:be,neg:y,tense:pr,pers:(_,3),nr:sg,pf:no,
```

```
    pg:no,headS:X,head:Y]><[X@is_a:someone,Y@tense:pr,Y@neg:y]).
==========================================================================
```

The learned LWFG rules after each iteration step:[3]

```
==========================================================================
The first iteration

Performance criterion - number of parsed E+ examples=49
Best set of rules
sbj(A><B-C)--->pro(D><B-C),{phi_comp(1,[A,D]),phi_onto(A)}.
sbj(A><B-C)--->pn(D><B-C),{phi_comp(2,[A,D]),phi_onto(A)}.
sbj(A><B-C)--->relpro(D><B-C),{phi_comp(3,[A,D]),phi_onto(A)} 3).
av0(A><B-C)--->sbj(D><B-E),aux(F><E-C),{phi_comp(4,[A,D,F]),phi_onto(A)}.
av0(A><B-C)--->aux(D><B-E),sbj(F><E-C),{phi_comp(5,[A,D,F]),phi_onto(A)}.
av0(A><B-C)--->relpro(D><B-E),aux(F><E-C),{phi_comp(6,[A,D,F]),phi_onto(A)}.
av4(A><B-C)--->av0(D><B-C),{phi_comp(7,[A,D]),phi_onto(A)}.
av4(A><B-C)--->av0(D><B-E),aux(F><E-C),{phi_comp(8,[A,D,F]),phi_onto(A)}.
av3(A><B-C)--->av0(D><B-C),{phi_comp(9,[A,D]),phi_onto(A)}.
av3(A><B-C)--->av0(D><B-E),aux(F><E-C),{phi_comp(10,[A,D,F]),phi_onto(A)}.
av2(A><B-C)--->av0(D><B-C),{phi_comp(11,[A,D]),phi_onto(A)}.
av2(A><B-C)--->av0(D><B-E),aux(F><E-C),{phi_comp(12,[A,D,F]),phi_onto(A)}.
av1(A><B-C)--->av0(D><B-C),{phi_comp(13,[A,D]),phi_onto(A)}.
av1(A><B-C)--->av0(D><B-E),aux(F><E-C),{phi_comp(14,[A,D,F]),phi_onto(A)}.
==========================================================================
The second iteration

Performance criterion - number of parsed E+ examples=68
Best set of rules:
sbj(A><B-C)--->pro(D><B-C),{phi_comp(1,[A,D]),phi_onto(A)}.
sbj(A><B-C)--->pn(D><B-C),{phi_comp(2,[A,D]),phi_onto(A)}.
sbj(A><B-C)--->relpro(D><B-C),{phi_comp(3,[A,D]),phi_onto(A)}.
av0(A><B-C)--->sbj(D><B-E),aux(F><E-C),{phi_comp(4,[A,D,F]),phi_onto(A)}.
av0(A><B-C)--->aux(D><B-E),sbj(F><E-C),{phi_comp(5,[A,D,F]),phi_onto(A)}.
av0(A><B-C)--->relpro(D><B-E),aux(F><E-C),{phi_comp(6,[A,D,F]),phi_onto(A)}.
av4(A><B-C)--->av3(D><B-C),{phi_comp(7,[A,D]),phi_onto(A)}.
av4(A><B-C)--->av3(D><B-E),aux(F><E-C),{phi_comp(8,[A,D,F]),phi_onto(A)}.
av3(A><B-C)--->av1(D><B-C),{phi_comp(9,[A,D]),phi_onto(A)}.
av3(A><B-C)--->av1(D><B-E),aux(F><E-C),{phi_comp(10,[A,D,F]),phi_onto(A)}.
av2(A><B-C)--->av1(D><B-C),{phi_comp(11,[A,D]),phi_onto(A)}.
av2(A><B-C)--->av1(D><B-E),aux(F><E-C),{phi_comp(12,[A,D,F]),phi_onto(A)}.
av1(A><B-C)--->av0(D><B-C),{phi_comp(13,[A,D]),phi_onto(A)}.
av1(A><B-C)--->av0(D><B-E),aux(F><E-C),{phi_comp(14,[A,D,F]),phi_onto(A)}.
==========================================================================
The third iteration:
```

[3] E+ is the representative sublanguage E_σ.

```
Performance criterion - number of parsed E+ examples=77
Best set of rules:
sbj(A><B-C)--->pro(D><B-C),{phi_comp(1,[A,D]),phi_onto(A)}.
sbj(A><B-C)--->pn(D><B-C),{phi_comp(2,[A,D]),phi_onto(A)}.
sbj(A><B-C)--->relpro(D><B-C),{phi_comp(3,[A,D]),phi_onto(A)}.
av0(A><B-C)--->sbj(D><B-E),aux(F><E-C),{phi_comp(4,[A,D,F]),phi_onto(A)}.
av0(A><B-C)--->aux(D><B-E),sbj(F><E-C),{phi_comp(5,[A,D,F]),phi_onto(A)}.
av0(A><B-C)--->relpro(D><B-E),aux(F><E-C),{phi_comp(6,[A,D,F]),phi_onto(A)}.
av4(A><B-C)--->av2(D><B-C),{phi_comp(7,[A,D]),phi_onto(A)}.
av4(A><B-C)--->av3(D><B-E),aux(F><E-C),{phi_comp(8,[A,D,F]),phi_onto(A)}.
av3(A><B-C)--->av2(D><B-C),{phi_comp(9,[A,D]),phi_onto(A)}.
av3(A><B-C)--->av2(D><B-E),aux(F><E-C),{phi_comp(10,[A,D,F]),phi_onto(A)}.
av2(A><B-C)--->av1(D><B-C),{phi_comp(11,[A,D]),phi_onto(A)}.
av2(A><B-C)--->av1(D><B-E),aux(F><E-C),{phi_comp(12,[A,D,F]),phi_onto(A)}.
av1(A><B-C)--->av0(D><B-C),{phi_comp(13,[A,D]),phi_onto(A)}.
av1(A><B-C)--->av0(D><B-E),aux(F><E-C),{phi_comp(14,[A,D,F]),phi_onto(A)}.
=========================================================================
The fourth iteration:

Performance criterion - number of parsed E+ examples=81
Best set of rules:
sbj(A><B-C)--->pro(D><B-C),{phi_comp(1,[A,D]),phi_onto(A)}.
sbj(A><B-C)--->pn(D><B-C),{phi_comp(2,[A,D]),phi_onto(A)}.
sbj(A><B-C)--->relpro(D><B-C),{phi_comp(3,[A,D]),phi_onto(A)}.
av0(A><B-C)--->sbj(D><B-E),aux(F><E-C),{phi_comp(4,[A,D,F]),phi_onto(A)}.
av0(A><B-C)--->aux(D><B-E),sbj(F><E-C),{phi_comp(5,[A,D,F]),phi_onto(A)}.
av0(A><B-C)--->relpro(D><B-E),aux(F><E-C),{phi_comp(6,[A,D,F]),phi_onto(A)}.
av4(A><B-C)--->av3(D><B-C),{phi_comp(7,[A,D]),phi_onto(A)}.
av4(A><B-C)--->av3(D><B-E),aux(F><E-C),{phi_comp(8,[A,D,F]),phi_onto(A)}.
av3(A><B-C)--->av2(D><B-C),{phi_comp(9,[A,D]),phi_onto(A)}.
av3(A><B-C)--->av2(D><B-E),aux(F><E-C),{phi_comp(10,[A,D,F]),phi_onto(A)}.
av2(A><B-C)--->av1(D><B-C),{phi_comp(11,[A,D]),phi_onto(A)}.
av2(A><B-C)--->av1(D><B-E),aux(F><E-C),{phi_comp(12,[A,D,F]),phi_onto(A)}.
av1(A><B-C)--->av0(D><B-C),{phi_comp(13,[A,D]),phi_onto(A)}.
av1(A><B-C)--->av0(D><B-E),aux(F><E-C),{phi_comp(14,[A,D,F]),phi_onto(A)}.
=========================================================================
```

A.2 Example of Grammar Learning Steps

In this appendix we exemplify the grammar learning steps that occur in learning the *ith* grammar rule from the *ith* representative example (see Procedure *Generate_Rule*$(\sigma, G', E_\sigma, K)$ and Figure 5.4 in section 5.1.2). For example, given the representative example *liking me today* (annotated with its semantic molecule), we

learn a grammar rule for reduced relative clauses post-modified by adjuncts.

```
================================================
cat:rrcl

OntoSeR=[~1@vft:ing,~1@is_a:like,~1@exp:~2,~1@perc:~3,~3@is_a:me,
        ~1@time:today]
w=[liking,me,today]]

Nchunk=2    [liking,me],[today]

Positive representative example:[[liking,me],[today]]

Annotation w' of representative example:
 [cat:rrcl,stype:no,ctype:s,vtype:norm,vft:nfin,vf:ing,val:tv,voice:act,
 head:A,mod:B]><[A@vft:ing,A@is_a:like,A@exp:B,A@perc:C,C@is_a:me,
 A@time:today].

Robparse semantics wi' of chunks :
(2, adv([cat:adv,head:A]><[A@time:today|B]-B)).
(2, adjc([link:A,atype:s,cat:adjc]><[A@time:today|B]-B)).
(1, nvo([head:A,headS:B,headC:C,cat:nvo,stype:no,ctype:s,vtype:norm,
        vft:nfin,vf:ing,val:tv,voice:act]><[A@vft:ing,A@is_a:like,A@exp:B,
        A@perc:C,C@is_a:me|D]-D)).
(1, rrcl([head:A,mod:B,stype:no,ctype:s,vtype:norm,vft:nfin,vf:ing,val:tv,
         voice:act,cat:rrcl]><[A@vft:ing,A@is_a:like,A@exp:B,A@perc:C,
         C@is_a:me|D]-D)).

Compositional constraint:
phi_comp(151,[A,C,D]):-
        eq(A,head:E,C,head:E),
        eq(A,head:F,D,head:F),
        eq(A,mod:G,C,headS:G),
        eq(A,stype:H,C,stype:H),
        eq(A,ctype:I,C,ctype:I),
        eq(A,vtype:J,C,vtype:J),
        eq(A,vft:nfin,C,vft:nfin),
        eq(A,vf:K,C,vf:K),
        eq(A,val:L,C,val:L),
        eq(A,voice:M,C,voice:M),
        eq(A,cat:rrcl),
        eq(C,cat:nvo),
        eq(D,cat:adv),
        e_list(A).

Most specific constraint rule:
rrcl(A><B-C)--->nvo(D><B-E),adv(F><E-C),{phi_comp(151,[A,D,F])}.
```

```
Generalization of the 1st right constituent

Compositional constraint:
phi_comp(151,[A,C,D]):-
        eq(A,head:E,C,head:E),
        eq(A,head:F,D,head:F),
        eq(A,mod:G,C,mod:G),
        eq(A,stype:H,C,stype:H),
        eq(A,ctype:I,C,ctype:I),
        eq(A,vtype:J,C,vtype:J),
        eq(A,vft:nfin,C,vft:nfin),
        eq(A,vf:K,C,vf:K),
        eq(A,val:L,C,val:L),
        eq(A,voice:M,C,voice:M),
        eq(A,cat:rrcl),
        eq(C,cat:rrcl),
        eq(D,cat:adv),
        e_list(A).

Performance criterion based on reduce semantics:
nvo=3 rrcl=4

Generalization of the 2nd right constituent

Compositional constraint:
phi_comp(151,[A,C,D]):-
        eq(A,head:E,C,head:E),
        eq(A,head:F,D,link:F),
        eq(A,mod:G,C,mod:G),
        eq(A,stype:H,C,stype:H),
        eq(A,ctype:I,C,ctype:I),
        eq(A,vtype:J,C,vtype:J),
        eq(A,vft:nfin,C,vft:nfin),
        eq(A,vf:K,C,vf:K),
        eq(A,val:L,C,val:L),
        eq(A,voice:M,C,voice:M),
        eq(A,cat:rrcl),
        eq(C,cat:rrcl),
        eq(D,cat:adjc),
        e_list(A).

Performance criterion based on reduce semantics:
adv=4 adjc=5

Best rule:
rrcl(A><B-C)--->rrcl(D><B-E),adjc(F><E-C),{phi_comp(151,[A,D,F])}.
========================================================================
```

A.3 Examples of Parsing/Generation

In this appendix we illustrate the reversibility of our grammar as presented in Section 3.5. It can be seen that the input to our robust parser is just the string, while the output is one or more syntagmas. The input to our generator is just the body of the semantic molecule, i.e., our semantic representation OntoSeR. The output of the generator is one or more syntagmas. In the examples given here, we show only the strings of these syntagmas if more are generated, except for the last syntagma which is chosen to illustrate the whole output form.

A.3.1 Finite Verbs

In this example we can see that for the string *who can not have been going* we obtained as output of parsing five syntagmas. They all have the same semantic representation, but they differ in their categories (and thus the heads of their semantic molecules are different). This string can be a wh-question (whcl), a relative clause (relcl), a finite clause (fcl), a clause with all its complements (svo), and a clause with subject and verb (sv). Since we only have one semantic representation, we have one input to our generator. We obtain three syntagmas, for which we show only their strings (we have three ways of verbalizing the modal and negation (*cannot, can't, can not*).

```
======================================================================
PARSING: string given as input:
w=[who,can,not,have,been,going]

Syntagmas resulted as output of parsing:
(w,w')=([who,can,not,have,been,going],
        [head:A,headS:B,stype:s,ctype:s,vtype:norm,vft:fin,val:iv,
        voice:act,int:y,dets:no,cat:whcl]><[B@is_a:who,A@mod:can,A@neg:y,
        A@bse:have,A@pf:y,A@pg:y,A@is_a:go,A@agt:B]).
```

```
(w,w')=([who,can,not,have,been,going],
        [head:C,mod:D,stype:s,ctype:s,vtype:norm,vft:fin,val:iv,
        voice:act,pers:(_,3),nr:_,int:no,dets:no,vf:no,cat:relcl]><
        [D@is_a:who,C@mod:can,C@neg:y,C@bse:have,C@pf:y,C@pg:y,C@is_a:go,
        C@agt:D]).

(w,w')=([who,can,not,have,been,going],
        [head:E,headS:F,stype:s,ctype:s,vtype:norm,vft:fin,val:iv,
        voice:act,pers:(_,3),nr:_,dets:no,int:_,cat:fcl]><[F@is_a:who,
        E@mod:can,E@neg:y,E@bse:have,E@pf:y,E@pg:y,E@is_a:go,E@agt:F]).

(w,w')=([who,can,not,have,been,going],
        [head:G,headS:H,stype:s,vtype:norm,vft:fin,vf:ing,val:iv,
        voice:act,int:_,dets:no,pers:(_,3),nr:_,ctype:s,cat:svo]><
        [H@is_a:who,G@mod:can,G@neg:y,G@bse:have,G@pf:y,G@pg:y,G@is_a:go,
        G@agt:H]).

(w,w')=([who,can,not,have,been,going],
        [head:I,headS:J,cat:sv,stype:s,vtype:norm,vft:fin,vf:ing,val:iv,
        aux:be,voice:act,int:_,dets:no,case:(n,ng),hum:y,neg:y,tense:mod,
        pers:(_,3),nr:_,pf:y,pg:y]><[J@is_a:who,I@mod:can,I@neg:y,
        I@bse:have,I@pf:y,I@pg:y,I@is_a:go,I@agt:J])]]).


GENERATION: OntoSeR given as input (in this case only one):
b=[~1@is_a:who,~2@mod:can,~2@neg:y,~2@bse:have,~2@pf:y,~2@pg:y,
  ~2@is_a:go,~2@agt:~1]

Strings of the syntagmas resulted as output of generation:
w=[who,cannot,have,been,going]
w=[who,can't,have,been,going]
w=[who,can,not,have,been,going]

Last syntagma from generation(output):
(w,w')=([who,can,not,have,been,going],
        [head:A,headS:B,stype:s,ctype:s,vtype:norm,vft:fin,val:iv,
        voice:act,int:y,dets:no,cat:whcl]><[B@is_a:who,A@mod:can,
        A@neg:y,A@bse:have,A@pf:y,A@pg:y,A@is_a:go,A@agt:B]).
=========================================================================
```

A.3.2 Relative Clause (without agreement)

In this appendix we present the case of parsing/generation of relative clauses in an experiment when we did not model the agreement between the head noun and the verb in the relative clause. Running the parser/generator pinpointed this error,

and let us refine the grammar by refining our representative examples.

```
==========================================================================
PARSING: string given as input:
w=[the,boy,who,loves,mary,tries,to,be,loved,by,her]

Syntagmas  resulted as output of parsing (in this case only one):
(w,w')=([the,boy,who,loves,mary,tries,to,be,loved,by,her],
        [head:A,headS:B,headC:C,stype:s,vtype:csbj,vft:fin,vf:no,val:tv,
        voice:act,dets:y,int:no,ctype:c,cat:fcl]><[B@det:the,B@is_a:boy,
        B@is_a:who,D@tense:pr,D@is_a:love,D@exp:B,D@perc:E,E@name:mary,
        A@tense:pr,A@is_a:try,A@ag:B,A@prop:C,C@vft:to,C@bse:be,
        C@voice:pas,C@is_a:love,C@exp:F,C@perc:B,exp@is_a:by,C@exp:F,
        F@is_a:her]).

GENERATION: OntoSeR given as input (in this case only one):
b=[~1@det:the,~1@is_a:boy,~1@is_a:who,~2@tense:pr,~2@is_a:love,~2@exp:~1,
  ~2@perc:~3,~3@name:mary,~4@tense:pr,~4@is_a:try,~4@ag:~1,~4@prop:~5,
  ~5@vft:to,~5@bse:be,~5@voice:pas,~5@is_a:love,~5@exp:~6,~5@perc:~1,
  exp@is_a:by,~5@exp:~6,~6@is_a:her]

Strings of the syntagmas resulted as output of generation:
w=[the,boy,who,love,mary,does,try,to,be,loved,by,her]
w=[the,boy,who,love,mary,tries,to,be,loved,by,her]
w=[the,boy,who,loves,mary,does,try,to,be,loved,by,her]
w=[the,boy,who,loves,mary,tries,to,be,loved,by,her]
w=[the,boy,who,do,love,mary,does,try,to,be,loved,by,her]
w=[the,boy,who,do,love,mary,tries,to,be,loved,by,her]
w=[the,boy,who,does,love,mary,does,try,to,be,loved,by,her]
w=[the,boy,who,does,love,mary,tries,to,be,loved,by,her]

Last syntagma from generation(output):
(w,w')=([the,boy,who,does,love,mary,tries,to,be,loved,by,her],
        [head:A,headS:B,headC:C,stype:s,vtype:csbj,vft:fin,vf:no,val:tv,
        voice:act,dets:y,int:no,ctype:c,cat:fcl]><[B@det:the,B@is_a:boy,
        B@is_a:who,D@tense:pr,D@is_a:love,D@exp:B,D@perc:E,E@name:mary,
        A@tense:pr,A@is_a:try,A@ag:B,A@prop:C,C@vft:to,C@bse:be,
        C@voice:pas,C@is_a:love,C@exp:F,C@perc:B,exp@is_a:by,C@exp:F,
        F@is_a:her]).
==========================================================================
```

A.3.3 Relative Clause (with agreement)

In this appendix we show the parsing/generation for relative clauses in our current grammar, where we model agreement.

```
=========================================================================
PARSING: string given as input:
w=[the,boy,who,loves,mary,tries,to,be,loved,by,her]

Syntagmas  resulted as output of parsing (in this case only one):
(w,w')=([the,boy,who,loves,mary,tries,to,be,loved,by,her],
        [head:A,headS:B,headC:C,stype:s,vtype:csbj,vft:fin,vf:no,val:tv,
        voice:act,pers:(_,3),nr:sg,dets:y,int:no,ctype:c,cat:fcl]><
        [B@det:the,B@is_a:boy,B@is_a:who,D@tense:pr,D@is_a:love,D@exp:B,
        D@perc:E,E@name:mary,A@tense:pr,A@is_a:try,A@ag:B,A@prop:C,
        C@vft:to,C@bse:be,C@voice:pas,C@is_a:love,C@exp:F,C@perc:B,
        exp@is_a:by,C@exp:F,F@is_a:her]).


GENERATION: OntoSeR given as input (in this case only one):
b=[~1@det:the,~1@is_a:boy,~1@is_a:who,~2@tense:pr,~2@is_a:love,~2@exp:~1,
  ~2@perc:~3,~3@name:mary,~4@tense:pr,~4@is_a:try,~4@ag:~1,~4@prop:~5,
  ~5@vft:to,~5@bse:be,~5@voice:pas,~5@is_a:love,~5@exp:~6,~5@perc:~1,
  exp@is_a:by,~5@exp:~6,~6@is_a:her]

Strings of the syntagmas resulted as output of generation:
w=[the,boy,who,loves,mary,does,try,to,be,loved,by,her]
w=[the,boy,who,loves,mary,tries,to,be,loved,by,her]
w=[the,boy,who,does,love,mary,does,try,to,be,loved,by,her]
w=[the,boy,who,does,love,mary,tries,to,be,loved,by,her]

Last syntagma from generation(output):
(w,w')=([the,boy,who,does,love,mary,tries,to,be,loved,by,her],
        [head:A,headS:B,headC:C,stype:s,vtype:csbj,vft:fin,vf:no,val:tv,
        voice:act,pers:(_,3),nr:sg,dets:y,int:no,ctype:c,cat:fcl]><
        [B@det:the,B@is_a:boy,B@is_a:who,D@tense:pr,D@is_a:love,D@exp:B,
        D@perc:E,E@name:mary,A@tense:pr,A@is_a:try,A@ag:B,A@prop:C,
        C@vft:to,C@bse:be,C@voice:pas,C@is_a:love,C@exp:F,C@perc:B,
        exp@is_a:by,C@exp:F,F@is_a:her]).
=========================================================================
```

A.3.4 Complex Utterance

In this appendix we present an example of parsing/generation for a complex sentence *King Abdullah has been working to try to get the Palestinian leaders to come to the table.*

```
=========================================================================
PARSING: string given as input:
w=[King Abdullah,has,been,working,to,try,to,get,the,palestinian,leaders,
   to,come,to,the,table]
```

```
Syntagmas  resulted as output of parsing (in this case only one):
(w,w')=(['King Abdullah',has,been,working,to,try,to,get,the,palestinian,
        leaders,to,come,to,the,table],
        [head:A,headS:B,headC:C,stype:s,vtype:csbj,vft:fin,val:tv,
        voice:act,pers:(_,3),nr:sg,dets:y,int:no,ctype:c,cat:fcl]><
        [B@name:'King Abdullah',A@tense:pr,A@pf:y,A@pg:y,A@is_a:work,
        A@ag:B,A@th:C,C@vft:to,C@is_a:try,C@ag:B,C@prop:D,D@vft:to,
        D@is_a:get,D@ag:B,D@th:E,D@src:F,E@det:the,G@is_a:palestinian,
        E@_:G,E@is_a:leaders,F@vft:to,F@is_a:come,F@ag:E,H@is_a:to,
        F@H:I,I@det:the,I@is_a:table]).

GENERATION: OntoSeR given as input (in this case only one):
b=[~1@name:King Abdullah,~2@tense:pr,~2@pf:y,~2@pg:y,~2@is_a:work,~2@ag:~1,
  ~2@th:~3,~3@vft:to,~3@is_a:try,~3@ag:~1,~3@prop:~4,~4@vft:to,~4@is_a:get,
  ~4@ag:~1,~4@th:~5,~4@src:~6,~5@det:the,~7@is_a:palestinian,~5@origin:~7,
  ~5@is_a:leaders,~6@vft:to,~6@is_a:come,~6@ag:~5,(mod-prop)@is_a:to,
  ~6@(mod-prop):~8,~8@det:the,~8@is_a:table]

Strings of the syntagmas resulted as output of generation:
w=[King Abdullah,has,been,working,to,try,to,get,the,palestinian,leaders,
   to,come,to,the,table]

Last syntagma from generation(output):
(w,w')=(['King Abdullah',has,been,working,to,try,to,get,the,palestinian,
        leaders,to,come,to,the,table],
        [head:A,headS:B,headC:C,stype:s,vtype:csbj,vft:fin,val:tv,
        voice:act,pers:(_,3),nr:sg,dets:y,int:no,ctype:c,cat:fcl]><
        [B@name:'King Abdullah',A@tense:pr,A@pf:y,A@pg:y,A@is_a:work,
        A@ag:B,A@th:C,C@vft:to,C@is_a:try,C@ag:B,C@prop:D,D@vft:to,
        D@is_a:get,D@ag:B,D@th:E,D@src:F,E@det:the,G@is_a:palestinian,
        E@origin:G,E@is_a:leaders,F@vft:to,F@is_a:come,F@ag:E,
        (mod-prop)@is_a:to,F@(mod-prop):H,H@det:the,H@is_a:table]).
=========================================================================
```

A.4 Samples of the Learned Grammar: Noun Compounds, Raising/Control, Reduced Relative Clauses

In this section we present fragments of our learned grammar that models noun compounds (see Section 6.1.2, Figure 6.10(b) and Section 7.1.2), raising and control (see Section 6.2 and Section 7.1.2), and reduced relative clauses (see Section

6.3.1 and Section 7.1.2). In Appendix A.4.1 we give samples of the representative examples for each of these phenomena, while in Appendix A.4.2 we give the learned grammar rules and samples of the learned compositional constraints.

A.4.1 Representative Examples

```
Sample of representative examples for noun compounds
====================================================================
pos([disease],[cat:na,nr:sg,head:X, mod:Y]><[X@is_a:disease, Y@P:X]).
pos([skin,disease],[cat:na,nr:sg,head:Y,mod:Z]><[X@is_a:skin, Y@P:X,
    Y@is_a:disease,Z@P1:Y]).
pos([disease],[cat:nc,modr:no,det:no,pers:(_,3),nr:sg,case:(_,ng),hum:no,
    gen:neutr,count:y,head:X]><[X@is_a:disease]).
pos([skin,disease],[cat:nc,modr:no,det:no,pers:(_,3),nr:sg,case:(_,ng),
    hum:no,gen:neutr,count:y,head:Y]><[X@is_a:skin, Y@P:X, Y@is_a:disease]).



Sample of representative examples for raising and control verbs
====================================================================

pos([he,tells,mary,to,love,me],[cat:fcl,stype:s,ctype:c,vtype:cobj,
    vft:fin,val:dv,voice:act,pers:(_,3),nr:sg,dets:y,int:no,head:B,
    headS:A,headC1:C,headC2:X]><[A@is_a:he,B@tense:pr,B@is_a:tell,B@ag:A,
    B@goal:C,B@prop:X,C@name:mary,X@vft:to,X@is_a:love,X@exp:C,X@perc:Z,
    Z@is_a:me]).
pos([he,seems,to,love,me],[cat:fcl,stype:s,ctype:c,vtype:rsbj,vft:fin,
    val:tv,voice:act,pers:(_,3),nr:sg,dets:y,int:no,head:B,headS:Y,
    headC:X]><[Y@is_a:he,B@tense:pr,B@is_a:seem,B@no_role:Y,B@prop:X,
    X@vft:to,X@is_a:love,X@exp:Y,X@perc:Z,Z@is_a:me]).
pos([he,tries,to,love,me],[cat:fcl,stype:s,ctype:c,vtype:csbj,vft:fin,
    val:tv,voice:act,pers:(_,3),nr:sg,dets:y,int:no,head:X1,headS:Y,
    headC:X]><[Y@is_a:he,X1@tense:pr,X1@is_a:try,X1@ag:Y,X1@prop:X,X@vft:to,
    X@is_a:love,X@exp:Y,X@perc:Z,Z@is_a:me]).

pos([he,promises,mary,to,love,me],[cat:fcl,stype:s,ctype:c,vtype:csbj,
    vft:fin,val:dv,voice:act,pers:(_,3),nr:sg,dets:y,int:no,head:X1,
    headS:Y,headC1:T,headC2:X]><[Y@is_a:he,X1@tense:pr,X1@is_a:promise,
    X1@ag:Y,X1@goal:T,X1@prop:X,T@name:mary,X@vft:to,X@is_a:love,X@exp:Y,
    X@perc:Z,Z@is_a:me]).


Sample of representative example for reduced relative clauses
====================================================================

pos([liking,me],[cat:rrcl,stype:no,ctype:s,vtype:norm,vft:nfin,vf:ing,
    val:tv,voice:act,head:A,mod:B]><[A@vft:ing,A@is_a:like,A@exp:B,A@perc:C,
```

```
    C@is_a:me]).
pos([loved,by,me],[cat:rrcl,stype:no,ctype:s,vtype:norm,vft:nfin,vf:ed,
    val:tv,voice:pas,head:A,mod:B]><[A@vft:ed,A@voice:pas,A@is_a:love,
    A@exp:C,A@perc:B,D@is_a:by,A@D:C,C@is_a:me]).
pos([loving,you,or,liking,me],[cat:rrcl,stype:no,ctype:s,vtype:norm,
    vft:nfin,vf:ing,val:tv,voice:act,head:X,mod:B]><[A@vft:ing,A@is_a:love,
    A@exp:B,A@perc:C,C@is_a:you,X:A or A1,A1@vft:ing,A1@is_a:like,A1@exp:B,
    A1@perc:C1,C1@is_a:me]).
pos([boy,liking,me],[cat:n,modr:no,det:no,pers:(_,3),nr:sg,case:(_,ng),
    hum:y,gen:male,count:y,head:B]><[B@is_a:boy,A@vft:ing,A@is_a:like,
    A@exp:B,A@perc:C,C@is_a:me]).
```

A.4.2 Learned Grammar Rules and Samples of Compositional Constraints

```
Learned grammar rules for noun phrase, raising and control verbs and
reduced relative clauses
==========================================================================

a(A><B-C)--->adj(D><B-C),{phi_comp(83,[A,D]),phi_onto(A)}.
a(A><B-C)--->adv(D><B-E),a(F><E-C),{phi_comp(84,[A,D,F]),phi_onto(A)}.
a(A><B-C)--->a(D><B-E),coord(F><E-G),a(H><G-C),{phi_comp(85,[A,D,F,H]),
            phi_onto(A)}.
na(A><B-C)--->noun(D><B-C),{phi_comp(86,[A,D]),phi_onto(A)}.
na(A><B-C)--->na(D><B-E),na(F><E-C),{phi_comp(87,[A,D,F]),phi_onto(A)}.
nc(A><B-C)--->noun(D><B-C),{phi_comp(88,[A,D]),phi_onto(A)}.
nc(A><B-C)--->na(D><B-E),nc(F><E-C),{phi_comp(89,[A,D,F]),phi_onto(A)}.
n(A><B-C)--->nc(D><B-C),{phi_comp(90,[A,D]),phi_onto(A)}.
n(A><B-C)--->a(D><B-E),n(F><E-C),{phi_comp(91,[A,D,F]),phi_onto(A)}.
n(A><B-C)--->det(D><B-E),n(F><E-C),{phi_comp(92,[A,D,F]),phi_onto(A)}.
n(A><B-C)--->pn(D><B-C),{phi_comp(93,[A,D]),phi_onto(A)}.
n(A><B-C)--->pro(D><B-C),{phi_comp(94,[A,D]),phi_onto(A)}.
n(A><B-C)--->n(D><B-E),coord(F><E-G),n(H><G-C),{phi_comp(95,[A,D,F,H]),
            phi_onto(A)}.
d(A><B-C)--->apro(D><B-C),{phi_comp(96,[A,D]),phi_onto(A)}.
d(A><B-C)--->n(D><B-E),iprep(F><E-C),{phi_comp(97,[A,D,F]),phi_onto(A)}.
d(A><B-C)--->det(D><B-E),d(F><E-C),{phi_comp(98,[A,D,F]),phi_onto(A)}.
d(A><B-C)--->d(D><B-E),coord(F><E-G),d(H><G-C),{phi_comp(99,[A,D,F,H]),
             phi_onto(A)}.
n(A><B-C)--->d(D><B-E),n(F><E-C),{phi_comp(100,[A,D,F]),phi_onto(A)}.
p(A><B-C)--->prep(D><B-C),{phi_comp(101,[A,D]),phi_onto(A)}.
p(A><B-C)--->p(D><B-E),coord(F><E-G),p(H><G-C),{phi_comp(102,[A,D,F,H]),
             phi_onto(A)}.
pc(A><B-C)--->p(D><B-E),n(F><E-C),{phi_comp(103,[A,D,F]),phi_onto(A)}.
pc(A><B-C)--->pc(D><B-E),coord(F><E-G),pc(H><G-C),{phi_comp(104,[A,D,F,H]),
             phi_onto(A)}.
n(A><B-C)--->n(D><B-E),pc(F><E-C),{phi_comp(105,[A,D,F]),phi_onto(A)}.
sbj(A><B-C)--->n(D><B-C),{phi_comp(106,[A,D]),phi_onto(A)}.
```

```
obj(A><B-C)--->n(D><B-C),{phi_comp(107,[A,D]),phi_onto(A)}.
pobj(A><B-C)--->pc(D><B-C),{phi_comp(108,[A,D]),phi_onto(A)}.

adjc(A><B-C)--->adv(D><B-C),{phi_comp(109,[A,D]),phi_onto(A)}.
adjc(A><B-C)--->pc(D><B-C),{phi_comp(110,[A,D]),phi_onto(A)}.
tocl(A><B-C)--->nvo(D><B-C),{phi_comp(111,[A,D]),phi_onto(A)}.
tocl(A><B-C)--->nvo(D><B-E),adjc(F><E-C),{phi_comp(112,[A,D,F]),phi_onto(A)}.
ncl(A><B-C)--->nv(D><B-E),obj(F><E-G),tocl(H><G-C),{phi_comp(113,[A,D,F,H]),
               phi_onto(A)}.
ncl(A><B-C)--->nv(D><B-E),tocl(F><E-C),{phi_comp(114,[A,D,F]),phi_onto(A)}.
ncl(A><B-C)--->nv(D><B-E),tocl(F><E-C),{phi_comp(115,[A,D,F]),phi_onto(A)}.
ncl(A><B-C)--->nv(D><B-E),obj(F><E-G),tocl(H><G-C),{phi_comp(116,[A,D,F,H]),
               phi_onto(A)}.
tocl(A><B-C)--->ncl(D><B-C),{phi_comp(117,[A,D]),phi_onto(A)}.
fcl(A><B-C)--->svo(D><B-C),{phi_comp(118,[A,D]),phi_onto(A)}.
fcl(A><B-C)--->fcl(D><B-E),adjc(F><E-C),{phi_comp(119,[A,D,F]),phi_onto(A)}.
fcl(A><B-C)--->adjc(D><B-E),fcl(F><E-C),{phi_comp(120,[A,D,F]),phi_onto(A)}.
fcl(A><B-C)--->sv(D><B-E),obj(F><E-G),tocl(H><G-C),{phi_comp(121,[A,D,F,H]),
               phi_onto(A)}.
fcl(A><B-C)--->sv(D><B-E),tocl(F><E-C),{phi_comp(122,[A,D,F]),phi_onto(A)}.
fcl(A><B-C)--->sv(D><B-E),tocl(F><E-C),{phi_comp(123,[A,D,F]),phi_onto(A)}.
fcl(A><B-C)--->sv(D><B-E),obj(F><E-G),tocl(H><G-C),{phi_comp(124,[A,D,F,H]),
               phi_onto(A)}.

rrcl(A><B-C)--->nvo(D><B-C),{phi_comp(146,[A,D]),phi_onto(A)}.
rrcl(A><B-C)--->nvo(D><B-C),{phi_comp(147,[A,D]),phi_onto(A)}.
rrcl(A><B-C)--->nvo(D><B-E),coord(F><E-G),rrcl(H><G-C),
               {phi_comp(148,[A,D,F,H]),phi_onto(A)}.
n(A><B-C)--->n(D><B-E),rrcl(F><E-C),{phi_comp(149,[A,D,F]),phi_onto(A)}.
rrcl(A><B-C)--->nvo(D><B-C),{phi_comp(150,[A,D]),phi_onto(A)}.
rrcl(A><B-C)--->rrcl(D><B-E),adjc(F><E-C),{phi_comp(151,[A,D,F]),phi_onto(A)}.


Sample of learned compositional constraints
===========================================

phi_comp(86, [A,C]) :-
        eq(A, head:D, C, head:D),
        eq(A, mod:E, C, mod:E),
        eq(A, nr:sg, C, nr:sg),
        eq(A, cat:na),
        eq(C, cat:noun),
        e_list(A).
phi_comp(87, [A,C,D]) :-
        eq(A, head:E, C, mod:E),
        eq(A, head:F, D, head:F),
        eq(A, mod:G, D, mod:G),
        eq(A, nr:sg, D, nr:sg),
        eq(C, nr:sg),
```

```
        eq(A, cat:na),
        eq(C, cat:na),
        eq(D, cat:na),
        e_list(A).
phi_comp(89, [A,C,D]) :-
        eq(A, head:E, C, mod:E),
        eq(A, head:F, D, head:F),
        eq(A, modr:G, D, modr:G),
        eq(A, det:H, D, det:H),
        eq(A, pers:(I,J), D, pers:(I,J)),
        eq(A, nr:K, D, nr:K),
        eq(A, case:(L,M), D, case:(L,M)),
        eq(A, hum:N, D, hum:N),
        eq(A, gen:O, D, gen:O),
        eq(A, count:P, D, count:P),
        eq(C, nr:sg),
        eq(A, cat:nc),
        eq(C, cat:na),
        eq(D, cat:nc),
        e_list(A).
phi_comp(92, [A,C,D]) :-
        eq(A, head:E, C, mod:E),
        eq(A, head:F, D, head:F),
        eq(A, modr:G, D, modr:G),
        eq(D, det:no),
        eq(A, pers:(H,I), D, pers:(H,I)),
        eq(A, nr:J, D, nr:J),
        eq(A, case:(K,L), D, case:(K,L)),
        eq(A, hum:M, D, hum:M),
        eq(A, gen:N, D, gen:N),
        eq(A, count:O, D, count:O),
        eq(A, nr:P, C, nr:P),
        eq(A, det:y),
        eq(A, cat:n),
        eq(C, cat:det),
        eq(D, cat:n),
        e_list((A,B)).

phi_comp(118, [A,C]) :-
        eq(A, head:D, C, head:D),
        eq(A, headS:E, C, headS:E),
        eq(A, stype:F, C, stype:F),
        eq(A, ctype:G, C, ctype:G),
        eq(A, vtype:H, C, vtype:H),
        eq(A, vft:fin, C, vft:fin),
        eq(A, val:I, C, val:I),
        eq(A, voice:J, C, voice:J),
        eq(A, pers:(K,L), C, pers:(K,L)),
        eq(A, nr:M, C, nr:M),
```

```
        eq(A, dets:N, C, dets:N),
        eq(A, int:O, C, int:O),
        eq(A, cat:fcl),
        eq(C, cat:svo),
        e_list(A).
phi_comp(119, [A,C,D]) :-
        eq(A, head:E, C, head:E),
        eq(A, head:F, D, link:F),
        eq(A, headS:G, C, headS:G),
        eq(A, stype:H, C, stype:H),
        eq(A, ctype:s, C, ctype:s),
        eq(A, vtype:I, C, vtype:I),
        eq(A, vft:fin, C, vft:fin),
        eq(A, val:J, C, val:J),
        eq(A, voice:K, C, voice:K),
        eq(A, pers:(L,M), C, pers:(L,M)),
        eq(A, nr:N, C, nr:N),
        eq(A, dets:O, C, dets:O),
        eq(A, int:P, C, int:P),
        eq(A, cat:fcl),
        eq(C, cat:fcl),
        eq(D, cat:adjc),
        e_list(A).
phi_comp(120, [A,C,D]) :-
        eq(A, head:E, C, link:E),
        eq(A, head:F, D, head:F),
        eq(A, headS:G, D, headS:G),
        eq(A, stype:H, D, stype:H),
        eq(A, ctype:I, D, ctype:I),
        eq(A, vtype:J, D, vtype:J),
        eq(A, vft:fin, D, vft:fin),
        eq(A, val:K, D, val:K),
        eq(A, voice:L, D, voice:L),
        eq(A, pers:(M,N), D, pers:(M,N)),
        eq(A, nr:O, D, nr:O),
        eq(A, dets:P, D, dets:P),
        eq(A, int:Q, D, int:Q),
        eq(A, cat:fcl),
        eq(C, cat:adjc),
        eq(D, cat:fcl),
        e_list(A).

phi_comp(148, [A,C,D,E]) :-
        eq(A, head:F, D, head:F),
        eq(A, mod:G, C, headS:G),
        eq(A, mod:H, E, mod:H),
        eq(C, head:I, D, link_l:I),
        eq(D, link_r:J, E, head:J),
        eq(A, stype:K, C, stype:K),
```

```
        eq(A, stype:L, E, stype:L),
        eq(A, ctype:M, C, ctype:M),
        eq(A, ctype:N, E, ctype:N),
        eq(A, vtype:O, C, vtype:O),
        eq(A, vtype:P, E, vtype:P),
        eq(A, vft:Q, C, vft:Q),
        eq(A, vft:R, E, vft:R),
        eq(A, vf:S, C, vf:S),
        eq(A, vf:T, E, vf:T),
        eq(A, val:U, C, val:U),
        eq(A, val:V, E, val:V),
        eq(A, voice:W, C, voice:W),
        eq(A, voice:X, E, voice:X),
        eq(A, cat:rrcl),
        eq(C, cat:nvo),
        eq(D, cat:c),
        eq(E, cat:rrcl),
        e_list(A).
phi_comp(149, [A,C,D]) :-
        eq(A, head:E, C, head:E),
        eq(A, head:F, D, mod:F),
        eq(A, modr:y),
        eq(C, modr:no),
        eq(A, det:G, C, det:G),
        eq(A, pers:(H,I), C, pers:(H,I)),
        eq(A, nr:J, C, nr:J),
        eq(A, case:(K,L), C, case:(K,L)),
        eq(A, hum:M, C, hum:M),
        eq(A, gen:N, C, gen:N),
        eq(A, count:O, C, count:O),
        eq(A, cat:n),
        eq(C, cat:n),
        eq(D, cat:rrcl),
        e_list(A).
phi_comp(151, [A,C,D]) :-
        eq(A, head:E, C, head:E),
        eq(A, head:F, D, link:F),
        eq(A, mod:G, C, mod:G),
        eq(A, stype:H, C, stype:H),
        eq(A, ctype:I, C, ctype:I),
        eq(A, vtype:J, C, vtype:J),
        eq(A, vft:nfin, C, vft:nfin),
        eq(A, vf:K, C, vf:K),
        eq(A, val:L, C, val:L),
        eq(A, voice:M, C, voice:M),
        eq(A, cat:rrcl),
        eq(C, cat:rrcl),
        eq(D, cat:adjc),
        e_list(A).
```

A.5 Long-Distance Dependencies in Wh-questions

In this appendix we give an example of processing long-distance dependencies in wh-questions (see Section 6.3.3 and Figure 6.26). For the input *What does the president seem to try to get from the farmers?* we show the output of parsing (before and after Φ_{onto} validation), the asserted text knowledge (TKR) and the asserted ontology-level knowledge (OKR).

```
========================================================================
>> Wh-Question (input)

w = [what,does,the,president,seem,to,try,to,get,from,the,farmers]


All Syntagmas (with inconsistent OntoSeR-) obtained as result of
parsing(output):

(w,w')=([what,does,the,president,seem,to,try,to,get,from,the,farmers],
        [head:A,headS:B,ans:C,stype:s,ctype:c,vtype:csbj,vft:fin,val:tv,
        voice:act,int:y,dets:y,cat:whcl]><[C@is_a:what,A@tense:pr,B@det:the,
        B@is_a:president,A@is_a:seem,A@no_ag:B,A@prop:D,D@vft:to,D@is_a:try,
        D@ag:B,D@prop:E,E@vft:to,E@is_a:get,E@ag:B,E@th:F,E@src:G,C=F,
        H@is_a:from,E@H:G,G@det:the,G@is_a:farmers]).              ***


(w,w')=([what,does,the,president,seem,to,try,to,get,from,the,farmers],
        [head:I,headS:J,ans:C,stype:s,ctype:c,vtype:csbj,vft:fin,val:tv,
        voice:act,int:y,dets:y,cat:whcl]><[C@is_a:what,I@tense:pr,J@det:the,
        J@is_a:president,I@is_a:seem,I@no_ag:J,I@prop:K,K@vft:to,K@is_a:try,
        K@ag:J,K@prop:L,L@vft:to,L@is_a:get,L@ag:J,L@th:M,L@src:N,C=N,C=M,
        O@is_a:from,L@O:P,P@det:the,P@is_a:farmers]).              *******


Uniques syntagma accepted by phi_onto with consistent OntoSeR+(output):

(w,w')=([what,does,the,president,seem,to,try,to,get,from,the,farmers],
        [head:A,headS:B,ans:C,stype:s,ctype:c,vtype:csbj,vft:fin,val:tv,
        voice:act,int:y,dets:y,cat:whcl]><[C@is_a:what,A@tense:pr,
        B@det:the,B@is_a:president,A@is_a:seem,A@no_ag:B,A@prop:D,D@vft:to,
        D@is_a:try,D@ag:B,D@prop:E,E@vft:to,E@is_a:get,E@ag:B,E@th:C,E@src:F,
        src@is_a:from,E@src:F,F@det:the,F@is_a:farmers]).           *

OntoSeR+
```

```
[C@is_a:what,A@tense:pr,B@det:the,B@is_a:president,A@is_a:seem,A@no_ag:B,
 A@prop:D,D@vft:to, D@is_a:try,D@ag:B,D@prop:E,E@vft:to,E@is_a:get,E@ag:B,
 E@th:C,E@src:F,src@is_a:from,E@src:F,F@det:the,F@is_a:farmers]

Asserted text knowledge:

TKR-query
~1@is_a::what.
~2@tense::pr.
~3@det::the.
~3@is_a::president.
~2@is_a::seem.
~2@no_ag::~3.
~2@prop::~4.
~4@vft::to.
~4@is_a::try.
~4@ag::~3.
~4@prop::~5.
~5@vft::to.
~5@is_a::get.
~5@ag::~3.
~5@th::~1.
~5@src::~6.
src@is_a::from.
~5@src::~6.
~6@det::the.
~6@is_a::farmers.

Asserted ontology knowledge:

OKR-query
#seem@prop:#:#try.
#try@ag:#:#president.
#try@prop:#:#get.
#get@ag:#:#president.
#get@th:#:#what.
#get@src:#:#farmers.
==========================================================================
```

A.6 Generating Representative Examples

In this appendix we present an example of generating the set of representative examples given a grammar G, and a sublanguage E_σ using *Find_Representative_Examples*(E_σ, G) and *Syntagma_Equivalence_Classes*(E_σ, G) algorithms presented in Section 3.6. We

show the results for the case of finite auxiliary verbs.

```
========================================================================
Input: Representative Sublanguage E+ (positive examples)
poss([who,can,have,been,being], _).
poss([has,someone,been,being], _).
poss([john,was,not], _).
poss([was,he,being], _).

Input: Grammar G
sbj(A><B-C)--->pro(D><B-C),{phi_comp(1,[A,D])}.
sbj(A><B-C)--->pn(D><B-C),{phi_comp(2,[A,D])}.
sbj(A><B-C)--->relpro(D><B-C),{phi_comp(3,[A,D])}.
av0(A><B-C)--->sbj(D><B-E),aux(F><E-C),{phi_comp(4,[A,D,F])}.
av0(A><B-C)--->aux(D><B-E),sbj(F><E-C),{phi_comp(5,[A,D,F])}.
av0(A><B-C)--->relpro(D><B-E),aux(F><E-C),{phi_comp(6,[A,D,F])}.
av1(A><B-C)--->av0(D><B-C),{phi_comp(7,[A,D])}.
av1(A><B-C)--->av0(D><B-E),aux(F><E-C),{phi_comp(8,[A,D,F])}.
av2(A><B-C)--->av1(D><B-C),{phi_comp(9,[A,D])}.
av2(A><B-C)--->av1(D><B-E),aux(F><E-C),{phi_comp(10,[A,D,F])}.
av3(A><B-C)--->av2(D><B-C),{phi_comp(11,[A,D])}.
av3(A><B-C)--->av2(D><B-E),aux(F><E-C),{phi_comp(12,[A,D,F])}.
av4(A><B-C)--->av3(D><B-C),{phi_comp(13,[A,D])}.
av4(A><B-C)--->av3(D><B-E),aux(F><E-C),{phi_comp(14,[A,D,F])}.

Output: Equivalence clases
eq_cll((aux,0), 0, ([have],1)).
eq_cll((aux,0), 0, ([have],1)).
eq_cll((aux,0), 0, ([being],1)).
eq_cll((av4,0), 14, ([who,can,have,been,being],5)).
eq_cll((aux,0), 0, ([being],1)).
eq_cll((av4,0), 13, ([who,can,have,been],4)).
eq_cll((av3,0), 12, ([who,can,have,been],4)).
eq_cll((aux,0), 0, ([been],1)).
eq_cll((av4,0), 13, ([who,can,have],3)).
eq_cll((av3,0), 11, ([who,can,have],3)).
eq_cll((av2,0), 10, ([who,can,have],3)).
eq_cll((aux,0), 0, ([have],1)).
eq_cll((av4,0), 13, ([who,can],2)).
eq_cll((av3,0), 11, ([who,can],2)).
eq_cll((av2,0), 9, ([who,can],2)).
eq_cll((av1,0), 7, ([who,can],2)).
eq_cll((av0,0), 6, ([who,can],2)).
eq_cll((aux,0), 0, ([can],1)).
eq_cll((sbj,0), 3, ([who],1)).
eq_cll((relpro,0), 0, ([who],1)).
eq_cll((av4,0), 14, ([has,someone,been,being],4)).
eq_cll((av4,0), 13, ([has,someone,been],3)).
eq_cll((av3,0), 12, ([has,someone,been],3)).
eq_cll((av4,0), 13, ([has,someone],2)).
```

```
eq_cll((av3,0), 11, ([has,someone],2)).
eq_cll((av2,0), 9, ([has,someone],2)).
eq_cll((av1,0), 7, ([has,someone],2)).
eq_cll((av0,0), 5, ([has,someone],2)).
eq_cll((sbj,0), 1, ([someone],1)).
eq_cll((pro,0), 0, ([someone],1)).
eq_cll((aux,0), 0, ([has],1)).
eq_cll((av4,0), 13, ([john,was,not],3)).
eq_cll((av3,0), 11, ([john,was,not],3)).
eq_cll((av2,0), 9, ([john,was,not],3)).
eq_cll((av1,0), 8, ([john,was,not],3)).
eq_cll((av4,0), 13, ([john,was],2)).
eq_cll((av3,0), 11, ([john,was],2)).
eq_cll((av2,0), 9, ([john,was],2)).
eq_cll((av1,0), 7, ([john,was],2)).
eq_cll((av0,0), 4, ([john,was],2)).
eq_cll((aux,0), 0, ([was],1)).
eq_cll((aux,0), 0, ([not],1)).
eq_cll((aux,0), 0, ([was],1)).
eq_cll((sbj,0), 2, ([john],1)).
eq_cll((pn,0), 0, ([john],1)).
eq_cll((av4,0), 14, ([was,he,being],3)).
eq_cll((av4,0), 13, ([was,he],2)).
eq_cll((av3,0), 11, ([was,he],2)).
eq_cll((av2,0), 9, ([was,he],2)).
eq_cll((av1,0), 7, ([was,he],2)).
eq_cll((av0,0), 5, ([was,he],2)).
eq_cll((sbj,0), 1, ([he],1)).
eq_cll((pro,0), 0, ([he],1)).


Output: Representative examples(strings and their semantic molecules)
(sbj,0),1,([he],[head:A,cat:sbj,stype:s,dets:y,pers:(_,3),nr:sg,
           case:(n,ng),hum:y]><[A@is_a:he]),1)).
(sbj,0),2,([john],[head:A,cat:sbj,stype:s,dets:y,pers:(_,3),nr:sg,
           case:(n,ng),hum:y]><[A@name:john]),1)).
(sbj,0),3,([who],[head:A,cat:sbj,stype:s,dets:no,pers:(_,3),nr:_,
          case:(n,ng),hum:y]><[A@is_a:who]),1)).
(av0,0),5,([was,he],[headS:A,head:B,cat:av0,stype:s,vtype:aux,vft:fin,
          int:y,dets:y,case:(n,ng),hum:y,aux:be,neg:no,tense:pt,
          pers:(_,3),nr:sg,pf:no,pg:no]><[B@tense:pt,A@is_a:he]),2)).
(av0,0),4,([john,was],[headS:A,head:B,cat:av0,stype:s,vtype:aux,
          vft:fin,int:no,dets:y,case:(n,ng),hum:y,aux:be,neg:no,tense:pt,
          pers:(_,3),nr:sg,pf:no,pg:no]><[A@name:john,B@tense:pt]),2)).
(av0,0),6,([who,can],[headS:A,head:B,cat:av0,stype:s,vtype:aux,vft:fin,
          int:_,dets:no,case:(n,ng),hum:y,aux:mod,neg:no,tense:mod,
          pers:(_,3),nr:_,pf:no,pg:no]><[A@is_a:who,B@mod:can]),2)).
(av1,0),7,([was,he],[headS:A,head:B,cat:av1,stype:s,vtype:aux,vft:fin,
          int:y,dets:y,case:(n,ng),hum:y,aux:be,neg:no,tense:pt,
```

```
          pers:(_,3),nr:sg,pf:no,pg:no]><[B@tense:pt,A@is_a:he]),2)).
(av1,0),8,([john,was,not],[headS:A,head:B,cat:av1,stype:s,vtype:aux,
          vft:fin,int:no,dets:y,case:(n,ng),hum:y,aux:be,neg:y,tense:pt,
          pers:(_,3),nr:sg,pf:no,pg:no]><[A@name:john,B@tense:pt,
          B@neg:y]),3)).
(av2,0),9,([was,he],[headS:A,head:B,cat:av2,stype:s,vtype:aux,vft:fin,
          int:y,dets:y,case:(n,ng),hum:y,aux:be,neg:no,tense:pt,pers:(_,3),
          nr:sg,pf:no,pg:no]><[B@tense:pt,A@is_a:he]),2)).
(av2,0),10,([who,can,have],[headS:A,head:B,cat:av2,stype:s,vtype:aux,
           vft:fin,int:_,dets:no,case:(n,ng),hum:y,aux:have,neg:no,
           tense:mod,pers:(_,3),nr:_,pf:no,pg:no]><[A@is_a:who,B@mod:can,
           B@bse:have]),3)).
(av3,0),11,([was,he],[headS:A,head:B,cat:av3,stype:s,vtype:aux,vft:fin,
           int:y,dets:y,case:(n,ng),hum:y,aux:be,neg:no,tense:pt,
           pers:(_,3),nr:sg,pf:no,pg:no]><[B@tense:pt,A@is_a:he]),2)).
(av3,0),12,([has,someone,been],[headS:A,head:B,cat:av3,stype:s,vtype:aux,
           vft:fin,int:y,dets:y,case:(n,ng),hum:_,aux:be,neg:no,tense:pr,
           pers:(_,3),nr:sg,pf:y,pg:no]><[B@tense:pr,A@is_a:someone,
           B@pf:y]),3)).
(av4,0),13,([was,he],[headS:A,head:B,cat:av4,stype:s,vtype:aux,vft:fin,
           int:y,dets:y,case:(n,ng),hum:y,aux:be,neg:no,tense:pt,
           pers:(_,3),nr:sg,pf:no,pg:no]><[B@tense:pt,A@is_a:he]),2)).
(av4,0),14,([was,he,being],[headS:A,head:B,cat:av4,stype:s,vtype:aux,
           vft:fin,int:y,dets:y,case:(n,ng),hum:y,aux:be,neg:no,tense:pt,
           pers:(_,3),nr:sg,pf:no,pg:y]><[B@tense:pt,A@is_a:he,B@pg:y]),
           3)).

Output: Grammar Gr:
sbj(A><B-C)--->pro(D><B-C),{phi_comp(1,[A,D])}.
sbj(A><B-C)--->pn(D><B-C),{phi_comp(2,[A,D])}.
sbj(A><B-C)--->relpro(D><B-C),{phi_comp(3,[A,D])}.
av0(A><B-C)--->aux(D><B-E),sbj(F><E-C),{phi_comp(5,[A,D,F])}.
av0(A><B-C)--->sbj(D><B-E),aux(F><E-C),{phi_comp(4,[A,D,F])}.
av0(A><B-C)--->relpro(D><B-E),aux(F><E-C),{phi_comp(6,[A,D,F])}.
av1(A><B-C)--->av0(D><B-C),{phi_comp(7,[A,D])}.
av1(A><B-C)--->av0(D><B-E),aux(F><E-C),{phi_comp(8,[A,D,F])}.
av2(A><B-C)--->av1(D><B-C),{phi_comp(9,[A,D])}.
av2(A><B-C)--->av1(D><B-E),aux(F><E-C),{phi_comp(10,[A,D,F])}.
av3(A><B-C)--->av2(D><B-C),{phi_comp(11,[A,D])}.
av3(A><B-C)--->av2(D><B-E),aux(F><E-C),{phi_comp(12,[A,D,F])}.
av4(A><B-C)--->av3(D><B-C),{phi_comp(13,[A,D])}.
av4(A><B-C)--->av3(D><B-E),aux(F><E-C),{phi_comp(14,[A,D,F])}.
=========================================================================
```

Appendix B

Example of an Ambiguous Utterance

In this appendix we present an example of parsing an ambiguous utterance *Monsanto's president who seems to try to get royalties from the farmers who grew genetically modified soya illegally will be giving a briefing for the Brazilian media tomorrow at noon.* We have two interpretations—one where *illegally* modifies the verb *get* and one where it modifies the verb *grow* (see also Section 6.4 and Figure 6.24, which shows the OKR representation of one of the interpretations).

```
======================================================================
>> Assertion
w= [Monsanto,'s,president,who,seems,to,try,to,get,royalties,from,the,
   farmers,who,grew,genetically,modified,soya,illegally,will,be,giving,a,
   briefing,for,the,brazilian,media,tomorrow,at,noon]

Syntagmas (OntoSeR-) returned by the parser:
(w,w')=(['Monsanto','\'s',president,who,seems,to,try,to,get,royalties,
        from,the,farmers,who,grew,genetically,modified,soya,illegally,will,
        be,giving,a,briefing,for,the,brazilian,media,tomorrow,at,noon],
        [head:A,headS:B,stype:s,ctype:s,vtype:norm,vft:fin,val:dv,voice:act,
        pers:(_,3),nr:sg,dets:y,int:no,cat:fcl]><[C@name:'Monsanto',B@of:C,
        B@is_a:president,B@is_a:who,D@tense:pr,D@is_a:seem,D@no_ag:B,
        D@prop:E,E@vft:to,E@is_a:try,E@ag:B,E@prop:F,F@vft:to,F@is_a:get,
        F@ag:B,F@th:G,F@src:H,G@is_a:royalties,I@is_a:from,F@I:H,H@det:the,
        H@is_a:farmers,H@is_a:who,J@tense:pt,J@is_a:grow,J@ag:H,J@th:K,
        L@viewpoint:genetically,L@is_a:modified,K@_:L,K@is_a:soya,
        F@manner:illegally,A@mod:will,A@bse:be,A@pg:y,A@is_a:give,A@ag:B,
        A@th:M,A@goal:N,M@det:a,M@is_a:briefing,O@is_a:for,A@O:N,N@det:the,
```

```
        P@is_a:brazilian,N@_:P,N@is_a:media,A@temporal:tomorrow,Q@is_a:at,
        A@Q:R,R@is_a:noon]).
(w,w')=(['Monsanto','\'s',president,who,seems,to,try,to,get,royalties,from,
        the,farmers,who,grew,genetically,modified,soya,illegally,will,be,
        giving,a,briefing,for,the,brazilian,media,tomorrow,at,noon],
        [head:S,headS:T,stype:s,ctype:s,vtype:norm,vft:fin,val:dv,voice:act,
        pers:(_,3),nr:sg,dets:y,int:no,cat:fcl]><[U@name:'Monsanto',T@of:U,
        T@is_a:president,T@is_a:who,V@tense:pr,V@is_a:seem,V@no_ag:T,
        V@prop:W,W@vft:to,W@is_a:try,W@ag:T,W@prop:X,X@vft:to,X@is_a:get,
        X@ag:T,X@th:Y,X@src:Z,Y@is_a:royalties,A1@is_a:from,X@A1:Z,
        Z@det:the,Z@is_a:farmers,Z@is_a:who,B1@tense:pt,B1@is_a:grow,
        B1@ag:Z,B1@th:C1,D1@viewpoint:genetically,D1@is_a:modified,C1@_:D1,
        C1@is_a:soya,B1@manner:illegally,S@mod:will,S@bse:be,S@pg:y,
        S@is_a:give,S@ag:T,S@th:E1,S@goal:F1,E1@det:a,E1@is_a:briefing,
        G1@is_a:for,S@G1:F1,F1@det:the,H1@is_a:brazilian,F1@_:H1,
        F1@is_a:media,S@temporal:tomorrow,I1@is_a:at,S@I1:J1,
        J1@is_a:noon])]]).


Number of syntagmas(OntoSeR-) returned by robust parser = 2


Syntagmas(OntoSeR+) accepted by phi_onto as consistent:
(w,b)=(['Monsanto','\'s',president,who,seems,to,try,to,get,royalties,
       from,the,farmers,who,grew,genetically,modified,soya,illegally,
       will,be,giving,a,briefing,for,the,brazilian,media,tomorrow,at,noon],
       [A@name:'Monsanto',B@of:A,B@is_a:president,B@is_a:who,C@tense:pr,
       C@is_a:seem,C@no_ag:B,C@prop:D,D@vft:to,D@is_a:try,D@ag:B,D@prop:E,
       E@vft:to,E@is_a:get,E@ag:B,E@th:F,E@src:G,F@is_a:royalties,
       src@is_a:from,E@src:G,G@det:the,G@is_a:farmers,G@is_a:who,H@tense:pt,
       H@is_a:grow,H@ag:G,H@th:I,J@viewpoint:genetically,J@is_a:modified,
       I@quality:J,I@is_a:soya,H@manner:illegally,K@mod:will,K@bse:be,
       K@pg:y,K@is_a:give,K@ag:B,K@th:L,K@goal:M,L@det:a,L@is_a:briefing,
       goal@is_a:for,K@goal:M,M@det:the,N@is_a:brazilian,M@origin:N,
       M@is_a:media,K@temporal:tomorrow,time@is_a:at,K@time:O,O@is_a:noon]).
(w,b)=(['Monsanto','\'s',president,who,seems,to,try,to,get,royalties,from,
       the,farmers,who,grew,genetically,modified,soya,illegally,will,
       be,giving,a,briefing,for,the,brazilian,media,tomorrow,at,noon],
       [P@name:'Monsanto',Q@of:P,Q@is_a:president,Q@is_a:who,R@tense:pr,
       R@is_a:seem,R@no_ag:Q,R@prop:S,S@vft:to,S@is_a:try,S@ag:Q,S@prop:T,
       T@vft:to,T@is_a:get,T@ag:Q,T@th:U,T@src:V,U@is_a:royalties,
       src@is_a:from,T@src:V,V@det:the,V@is_a:farmers,V@is_a:who,W@tense:pt,
       W@is_a:grow,W@ag:V,W@th:X,Y@viewpoint:genetically,Y@is_a:modified,
       X@quality:Y,X@is_a:soya,T@manner:illegally,Z@mod:will,Z@bse:be,
       Z@pg:y,Z@is_a:give,Z@ag:Q,Z@th:A1,Z@goal:B1,A1@det:a,
       A1@is_a:briefing,goal@is_a:for,Z@goal:B1,B1@det:the,
       C1@is_a:brazilian,B1@origin:C1,B1@is_a:media,Z@temporal:tomorrow,
       time@is_a:at,Z@time:D1,D1@is_a:noon]]]).
```

```
Number of syntagmas (OntoSeR+) validated by phi_onto = 2


1.First alternative (modifies the verb 'get')
====================
Asserted text knowledge:

TKR-assert
~4@name ':a:' 'Monsanto'.
~5@of ':a:'~4.
~5@is_a ':a:' president.
~5@is_a ':a:' who.
~6@tense ':a:' pr.
~6@is_a ':a:' seem.
~6@no_ag ':a:'~5.
~6@prop ':a:'~7.
~7@vft ':a:' to.
~7@is_a ':a:' try.
~7@ag ':a:'~5.
~7@prop ':a:'~8.
~8@vft ':a:' to.
~8@is_a ':a:' get.
~8@ag ':a:'~5.
~8@th ':a:'~9.
~8@src ':a:'~10.
~9@is_a ':a:' royalties.
src@is_a ':a:' from.
~8@src ':a:'~10.
~10@det ':a:' the.
~10@is_a ':a:' farmers.
~10@is_a ':a:' who.
~11@tense ':a:' pt.
~11@is_a ':a:' grow.
~11@ag ':a:'~10.
~11@th ':a:'~12.
~13@viewpoint ':a:' genetically.
~13@is_a ':a:' modified.
~12@quality ':a:'~13.
~12@is_a ':a:' soya.
~8@manner ':a:' illegally. ************
~14@mod ':a:' will.
~14@bse ':a:' be.
~14@pg ':a:' y.
~14@is_a ':a:' give.
~14@ag ':a:'~5.
~14@th ':a:'~15.
~14@goal ':a:'~16.
~15@det ':a:' a.
~15@is_a ':a:' briefing.
```

```
goal@is_a ':a:' for.
~14@goal ':a:'~16.
~16@det ':a:' the.
~17@is_a ':a:' brazilian.
~16@origin ':a:'~17.
~16@is_a ':a:' media.
~14@temporal ':a:' tomorrow.
time@is_a ':a:' at.
~14@time ':a:'~18.
~18@is_a ':a:' noon.

Asserted ontology knowledge:

OKR-assert
#concept@sub:=:#concept.
#concept@sub:=:#person.
#person@sub:=:#president.
#person@sub:=:#farmer.
#president@of:=:#'Monsanto'.
#seem^6@prop:=:#try^7.
#try^7@ag:=:#president.
#try^7@prop:=:#get^8.
#get^8@ag:=:#president.
#get^8@th:=:#royalties.
#get^8@src:=:#farmers^10.
#get^8@manner:=:illegally. ************
#grow^11@ag:=:#farmers^10.
#grow^11@th:=:#soya^12.
#modified^13@viewpoint:=:genetically.
#soya^12@quality:=:#modified^13.
#give^14@mod:=:will.
#give^14@ag:=:#president.
#give^14@th:=:#briefing.
#give^14@goal:=:#media^16.
#give^14@temporal:=:tomorrow.
#give^14@time:=:#noon.
#media^16@origin:=:#brazilian.


2.Second alternative (modifies the verb 'grow')
===================

Asserted text knowledge:

TKR-assert
~4@name ':a:' 'Monsanto'.
~5@of ':a:'~4.
~5@is_a ':a:' president.
~5@is_a ':a:' who.
```

```
~6@tense ':a:' pr.
~6@is_a ':a:' seem.
~6@no_ag ':a:'~5.
~6@prop ':a:'~7.
~7@vft ':a:' to.
~7@is_a ':a:' try.
~7@ag ':a:'~5.
~7@prop ':a:'~8.
~8@vft ':a:' to.
~8@is_a ':a:' get.
~8@ag ':a:'~5.
~8@th ':a:'~9.
~8@src ':a:'~10.
~9@is_a ':a:' royalties.
src@is_a ':a:' from.
~8@src ':a:'~10.
~10@det ':a:' the.
~10@is_a ':a:' farmers.
~10@is_a ':a:' who.
~11@tense ':a:' pt.
~11@is_a ':a:' grow.
~11@ag ':a:'~10.
~11@th ':a:'~12.
~13@viewpoint ':a:' genetically.
~13@is_a ':a:' modified.
~12@quality ':a:'~13.
~12@is_a ':a:' soya.
~11@manner ':a:' illegally. ************
~14@mod ':a:' will.
~14@bse ':a:' be.
~14@pg ':a:' y.
~14@is_a ':a:' give.
~14@ag ':a:'~5.
~14@th ':a:'~15.
~14@goal ':a:'~16.
~15@det ':a:' a.
~15@is_a ':a:' briefing.
goal@is_a ':a:' for.
~14@goal ':a:'~16.
~16@det ':a:' the.
~17@is_a ':a:' brazilian.
~16@origin ':a:'~17.
~16@is_a ':a:' media.
~14@temporal ':a:' tomorrow.
time@is_a ':a:' at.
~14@time ':a:'~18.
~18@is_a ':a:' noon.

Asserted ontology knowledge:
```

```
OKR-assert
#concept@sub:=:#concept.
#concept@sub:=:#person.
#person@sub:=:#president.
#person@sub:=:#farmer.
#president@of:=:#'Monsanto'.
#seem^6@prop:=:#try^7.
#try^7@ag:=:#president.
#try^7@prop:=:#get^8.
#get^8@ag:=:#president.
#get^8@th:=:#royalties.
#get^8@src:=:#farmers^10.
#grow^11@ag:=:#farmers^10.
#grow^11@th:=:#soya^12.
#grow^11@manner:=:illegally. ***********
#modified^13@viewpoint:=:genetically.
#soya^12@quality:=:#modified^13.
#give^14@mod:=:will.
#give^14@ag:=:#president.
#give^14@th:=:#briefing.
#give^14@goal:=:#media^16.
#give^14@temporal:=:tomorrow.
#give^14@time:=:#noon.
#media^16@origin:=:#brazilian.
==========================================================================
```

Appendix C

Acquisition and Querying of a Pilot OKR-annotated treebank

In this appendix we present the entire pilot experiment described in Section 8.4.

C.1 Medical Definitions

In this appendix we give the 17 medical definitions used in this pilot experiments together with the number of different syntagmas obtained without and with semantic validation (Φ_{onto}).

```
========================================================================
>> Assertion= [hepatitis,is,a,disease,that,inflames,the,liver]
Number of syntagmas (OntoSeR-) from robust parser = 1
Number of syntagmas (OntoSeR+) validated by phi_onto = 1

>> Assertion= [hepatitis,is,a,disease,caused,by,infectious,or,toxic,agents,
              and, characterized,by,jaundice,',',fever,and,liver,enlargement]
Number of syntagmas (OntoSeR-) from robust parser = 2
Number of syntagmas (OntoSeR+) validated by phi_onto = 1

>> Assertion= ['Hepatitis A',is,an,acute,but,benign,viral,hepatitis,caused,
              by,a,virus,that,does,not,persist,in,the,blood,serum]
Number of syntagmas (OntoSeR-) from robust parser = 1
Number of syntagmas (OntoSeR+) validated by phi_onto = 1

>> Assertion= ['Hepatitis B',is,an,acute,viral,hepatitis,caused,by,a,virus,
              that,tends,to,persist,in,the,blood,serum]
Number of syntagmas (OntoSeR-) from robust parser = 1
```

```
Number of syntagmas (OntoSeR+) validated by phi_onto = 1

>> Assertion= [acne,is,an,inflammatory,skin,disease,characterized,by,
              pimples,that,can, appear,on,any,part,of,the,body]
Number of syntagmas (OntoSeR-) from robust parser = 2
Number of syntagmas (OntoSeR+) validated by phi_onto = 2

>> Assertion= [acne,is,a,skin,disease,caused,by,overactive,oil,glands]
Number of syntagmas (OntoSeR-) from robust parser = 1
Number of syntagmas (OntoSeR+) validated by phi_onto = 1

>> Assertion= [acne,is,an,inflammatory,disease,involving,the,sebaceous,
              glands,of,the, skin]
Number of syntagmas (OntoSeR-) from robust parser = 1
Number of syntagmas (OntoSeR+) validated by phi_onto = 1

>> Assertion= [acne,is,characterized,by,papules,or,pustules,or,comedones]
Number of syntagmas (OntoSeR-) from robust parser = 4
Number of syntagmas (OntoSeR+) validated by phi_onto = 1

>> Assertion= [acne,is,a,skin,disease,characterized,by, papules,and,
              pustules,on,the,face,and,neck]
Number of syntagmas (OntoSeR-) from robust parser = 2
Number of syntagmas (OntoSeR+) validated by phi_onto = 2

>> Assertion= ['Addison''s disease',is,a,degenerative,disease,caused,by,
              a,deficiency,in,adrenocortical,hormones,and,characterized,by,
              a,'bronze-like',pigmentation,of,the,skin,and,a,low,blood,
              pressure]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 3

>> Assertion= ['Addison''s disease',is,a,rare,disorder,caused,by,a,
              deficiency,of,hydrocortisone,and,characterized,by,
              anemia,',',weight,loss,and,extreme,weakness]
Number of syntagmas (OntoSeR-) from robust parser = 2
Number of syntagmas (OntoSeR+) validated by phi_onto = 1

>> Assertion= [cataract,is,an,eye,disease,characterized,by,partial,or,
              complete,opacity,on,or,in,the,lens,or,capsule]
Number of syntagmas (OntoSeR-) from robust parser = 1
Number of syntagmas (OntoSeR+) validated by phi_onto = 1

>> Assertion= [cataract,is,caused,by,eye,lens,opacity,impairing,vision,
              or,causing,blindness]
Number of syntagmas (OntoSeR-) from robust parser = 4
Number of syntagmas (OntoSeR+) validated by phi_onto = 2

>> Assertion= ['Acute otitis media',is,a,disorder,involving,inflammation,
```

```
              and,infection,of,the,structures,of,the,middle,ear]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 3

>> Assertion= ['Acute otitis media',is,characterized,by,an,inflammation,of,
              the,middle,ear,with,signs,of,infection]
Number of syntagmas (OntoSeR-) from robust parser = 9
Number of syntagmas (OntoSeR+) validated by phi_onto = 8

>> Assertion= ['Acute otitis media',is,caused,by,a,bacterial,or,viral,
              infection,of,the,middle,ear]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 2

>> Assertion= ['endocrine glands',are,glands,that,secrete,substances,
              which,influence,metabolism,and,body,functions]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 3
==========================================================================
```

C.2 Acquired Pilot OKR-annotated treebank

In this appendix we give the entire semi-automatically acquired OKR-annotated treebank. The concept *#concept* is the top element of the hierarchy, and it is a-priori given (see also Figure C.1 for a visualization of this entire pilot OKR-annotated treebank).

```
==========================================================================
#concept@sub:=:#concept.
#concept@sub:=:#disease.
#disease@sub:=:#hepatitis.
#inflame^3@exp:=:#liver.
#cause^6@ag:=:#agents^7.
#agents^7@type_of:=:#infectious.
#agents^7@kind_of:=:#toxic.
#characterize^12@ag:=:[#jaundice,',',#fever,and,#enlargement^17].
#enlargement^17@body_part:=:#liver.
#hepatitis@sub:=:#'Hepatitis A'.
#'Hepatitis A'@duration:=:#acute.
#'Hepatitis A'@benignity:=:#benign.
#'Hepatitis A'@kind_of:=:#viral.
#cause^24@ag:=:#virus^25.
#persist^26@neg:=:y.
#persist^26@th:=:#virus^25.
```

```
#persist^26@loc_int:=:#serum^27.
#serum^27@of:=:#blood.
#hepatitis@sub:=:#'Hepatitis B'.
#'Hepatitis B'@duration:=:#acute.
#'Hepatitis B'@kind_of:=:#viral.
#cause^32@ag:=:#virus^33.
#tend^34@prop:=:#persist^35.
#persist^35@th:=:#virus^33.
#persist^35@loc_int:=:#serum^27.
#disease@sub:=:#acne.
#acne@kind_of:=:#inflammatory.
#acne@body_part:=:#skin.
#characterize^41@ag:=:#pimples^42.
#appear^43@mod:=:can.
#appear^43@th:=:#pimples^42.
#appear^43@loc_ext:=:#part^44.
#part^44@of:=:#body.
#cause^48@ag:=:#glands^49.
#glands^49@intensity:=:#overactive.
#glands^49@kind_of:=:#oil.
#involve^54@perc:=:#glands^55.
#glands^55@kind_of:=:#sebaceous.
#glands^55@of:=:#skin.
#characterize^59@ag:=:[#papules,or,#pustules,or,#comedones].
#characterize^67@ag:=:[#papules^69,and,#pustules^70].
[#papules^69,and,#pustules^70]@loc_ext:=:[#face,and,#neck].
#concept@sub:=:#disorder.
#disease@sub:=:#'Addison\'s disease'.
#'Addison\'s disease'@severity:=:#degenerative.
#cause^78@ag:=:#deficiency^79.
#deficiency^79@loc_int:=:#hormones^80.
#hormones^80@kind_of:=:#adrenocortical.
#characterize^83@ag:=:[#pigmentation^85,and,#pressure^88].
#pigmentation^85@color:=:#'bronze-like'.
#pigmentation^85@of:=:#skin.
#pressure^88@level:=:#low.
#pressure^88@of:=:#blood.
#disorder@sub:=:#'Addison\'s disease'.
#'Addison\'s disease'@commonality:=:#rare.
#cause^93@ag:=:#deficiency^94.
#deficiency^94@of:=:#hydrocortisone.
#characterize^97@ag:=:[#anemia,',',#loss^102,and,#weakness^103].
#loss^102@of:=:#weight.
#weakness^103@degree:=:#extreme.
#disease@sub:=:#cataract.
#cataract@body_part:=:#eye.
#characterize^108@ag:=:#opacity^109.
#opacity^109@degree:=:#partial.
#opacity^109@completeness:=:#complete.
```

```
#opacity^109@loc_ext:=:[#lens,or,#capsule].
#opacity^109@loc_int:=:[#lens,or,#capsule].
#cause^118@ag:=:#opacity^119.
#lens^121@body_part:=:#eye.
#opacity^119@of:=:#lens^121.
#impair^122@ag:=:#opacity^119.
#impair^122@exp:=:#vision.
#cause^125@ag:=:#opacity^119.
#cause^125@th:=:#blindness.
#concept@sub:=:#infection.
#disorder@sub:=:#'Acute otitis media'.
#involve^130@perc:=:[#inflammation^132,and,#infection].
[#inflammation^132,and,#infection]@of:=:#structures^134.
#structures^134@of:=:#ear^135.
#ear^135@position:=:#middle.
#characterize^138@ag:=:#inflammation^139.
#characterize^138@concomitant:=:#signs^142.
#signs^142@of:=:#infection.
#infection@kind_of:=:#bacterial.
#infection@kind_of:=:#viral.
#concept@sub:=:#glands.
#glands@sub:=:#'endocrine glands'.
#secrete^154@emission:=:#substances^155.
#influence^156@ag:=:#substances^155.
#influence^156@th:=:[#metabolism,and,#functions^159].
#functions^159@of:=:#body.
#inflammation^139@of:=:#ear^135.
#infection@of:=:#ear^135.
#inflame^3@ag:=:#hepatitis.
#cause^6@th:=:#hepatitis.
#characterize^12@th:=:#hepatitis.
#cause^24@th:=:#'Hepatitis A'.
#cause^32@th:=:#'Hepatitis B'.
#characterize^41@th:=:#acne.
#cause^48@th:=:#acne.
#involve^54@exp:=:#acne.
#characterize^59@th:=:#acne.
#characterize^67@th:=:#acne.
#cause^78@th:=:#'Addison\'s disease'.
#characterize^83@th:=:#'Addison\'s disease'.
#cause^93@th:=:#'Addison\'s disease'.
#characterize^97@th:=:#'Addison\'s disease'.
#characterize^108@th:=:#cataract.
#cause^118@th:=:#cataract.
#involve^130@exp:=:#'Acute otitis media'.
#characterize^138@th:=:#'Acute otitis media'.
#cause^145@ag:=:#infection.
#cause^145@th:=:#'Acute otitis media'.
#secrete^154@src:=:#'endocrine glands'.
```

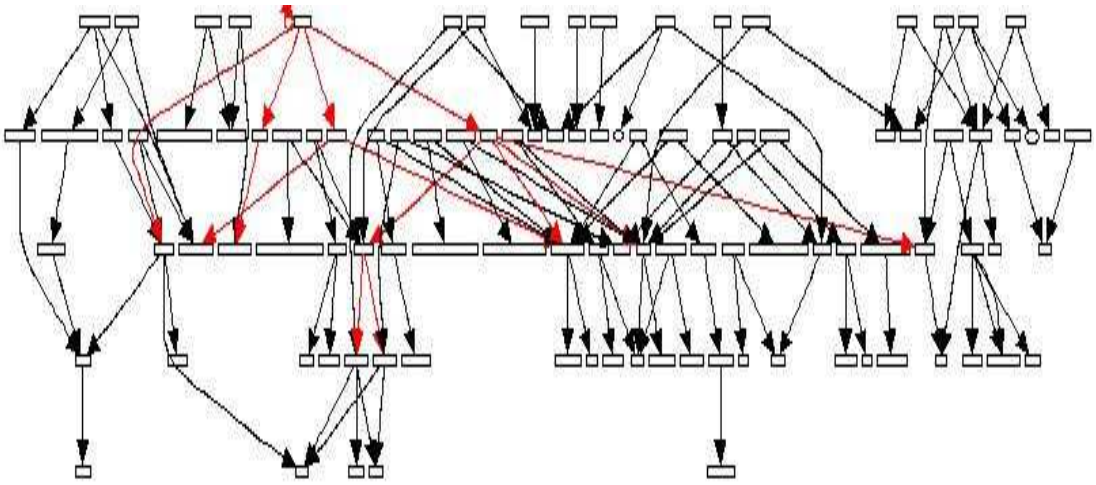


Figure C.1: OKR-annotated treebank obtained from 17 medical definitions

==========

C.3 NL-Querying Experiment

In this appendix we present the output of our NL-querying experiment. We created a set of 29 questions, both precise (22) and vague (7). The answers that are obtained are at the concept level. In this appendix we give the short answer (just the concept itself), while in the next appendix we give samples of full concept-level answers which takes into account the entire context of the concept. The answers are at the concept level because we do not have reversibility from the OKR, and obtaining natural language answers will require text generation, which is outside the scope of this dissertation. In this appendix we also show the ambiguity of questions due to our treatment of long distance dependencies, by showing the number of syntagmas that we obtain without and with semantic validation.

```
==========================================================================
>> Question= [which,are,viral,diseases]
Number of syntagmas (OntoSeR-) from robust parser = 1
```

```
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #'Hepatitis A'
>> Answer = #'Hepatitis B'

>> Question= [which,are,inflammatory,diseases]
Number of syntagmas (OntoSeR-) from robust parser = 1
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #acne

>> Question= [which,are,degenerative,diseases]
Number of syntagmas (OntoSeR-) from robust parser = 1
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #'Addison\'s disease'

>> Question= [which,are,rare,disorders]
Number of syntagmas (OntoSeR-) from robust parser = 1
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #'Addison\'s disease'

>> Question= [what,is,caused,by,infectious,or,toxic,agents]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #hepatitis

>> Question= [what,is,characterized,by,jaundice,and,liver,enlargement]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #hepatitis

>> Question= [what,causes,hepatitis]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #agents^7

>> Question= [what,is,caused,by,a,virus,that,does,not,persist,in,the,
             blood,serum]
Number of syntagmas (OntoSeR-) from robust parser = 5
Number of syntagmas (OntoSeR+) validated by phi_onto = 2
>> Answer = #'Hepatitis A'

>> Question= [what,is,caused,by,a,virus,that,persists,in,the,blood,serum]
Number of syntagmas (OntoSeR-) from robust parser = 5
Number of syntagmas (OntoSeR+) validated by phi_onto = 2
>> Answer = #'Hepatitis B'

>> Question= [what,is,caused,by,something,that,does,not,persist,in,the,
             blood,serum]
Number of syntagmas (OntoSeR-) from robust parser = 5
Number of syntagmas (OntoSeR+) validated by phi_onto = 2
```

```
>> Answer = #'Hepatitis A'

>> Question= [what,is,characterized,by,pustules,on,the,face]
Number of syntagmas (OntoSeR-) from robust parser = 5
Number of syntagmas (OntoSeR+) validated by phi_onto = 2
>> Answer = #acne

>> Question= [what,causes,acne]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #glands^49

>> Question= [what,does,acne,involve]
Number of syntagmas (OntoSeR-) from robust parser = 1
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #glands^55

>> Question= [what,characterizes,acne]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #pimples^42
>> Answer = [#papules,or,#pustules,or,#comedones]
>> Answer = [#papules^69,and,#pustules^70]

>> Question= [what,is,characterized,by,something,on,the,face]
Number of syntagmas (OntoSeR-) from robust parser = 5
Number of syntagmas (OntoSeR+) validated by phi_onto = 2
>> Answer = #acne

>> Question= [what,causes,'Addison''s disease']
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #deficiency^79
>> Answer = #deficiency^94

>> Question= [what,characterizes,'Addison''s disease']
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = [#pigmentation^85,and,#pressure^88]
>> Answer = [#anemia,,,#loss^102,and,#weakness^103]

>> Question= [what,is,characterized,by,pigmentation,of,the,skin]
Number of syntagmas (OntoSeR-) from robust parser = 5
Number of syntagmas (OntoSeR+) validated by phi_onto = 2
>> Answer = #'Addison\'s disease'

>> Question= [what,is,caused,by,something,impairing,vision]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
```

```
>> Answer = #blindness
>> Answer = #cataract

>> Question= [what,characterizes,cataract]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #opacity^109

>> Question= [what,causes,'Acute otitis media']
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #infection

>> Question= [what,characterizes,'Acute otitis media']
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #inflammation^139

>> Question= [what,involve,inflammation,and,infection,of,the,structures,
             of,the,middle,ear]
Number of syntagmas (OntoSeR-) from robust parser = 17
Number of syntagmas (OntoSeR+) validated by phi_onto = 6
>> Answer = #'Acute otitis media'

>> Question= [what,secrete,substances,which,influence,metabolism]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #'endocrine glands'

>> Question= [what,secrete,substances,which,influence,body,functions]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #'endocrine glands'

>> Question= [what,is,caused,by,something]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #blindness
>> Answer = #hepatitis
>> Answer = #'Hepatitis A'
>> Answer = #'Hepatitis B'
>> Answer = #acne
>> Answer = #'Addison\'s disease'
>> Answer = #'Addison\'s disease'
>> Answer = #cataract
>> Answer = #'Acute otitis media'

>> Question= [what,causes,something]
Number of syntagmas (OntoSeR-) from robust parser = 3
```

```
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #agents^7
>> Answer = #virus^25
>> Answer = #virus^33
>> Answer = #glands^49
>> Answer = #deficiency^79
>> Answer = #deficiency^94
>> Answer = #opacity^119
>> Answer = #opacity^119
>> Answer = #infection

>> Question= [what,is,characterized,by,something]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = #hepatitis
>> Answer = #acne
>> Answer = #acne
>> Answer = #acne
>> Answer = #'Addison\'s disease'
>> Answer = #'Addison\'s disease'
>> Answer = #cataract
>> Answer = #'Acute otitis media'

>> Question= [what,characterizes,something]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
>> Answer = [#jaundice,,,#fever,and,#enlargement^17]
>> Answer = #pimples^42
>> Answer = [#papules,or,#pustules,or,#comedones]
>> Answer = [#papules^69,and,#pustules^70]
>> Answer = [#pigmentation^85,and,#pressure^88]
>> Answer = [#anemia,,,#loss^102,and,#weakness^103]
>> Answer = #opacity^109
>> Answer = #inflammation^139
========================================================================
```

C.4 Samples of Full Concept-level Answers to NL-Queries

```
========================================================================
>> Question= [what,is,caused,by,something]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
OKR-query
#cause@ag:#:#something.
#cause@th:#:#what.
```

```
>> Answer = #blindness

#cause^125@ag:==:#opacity^119.
#lens^121@body_part:==:#eye.
#opacity^119@of:==:#lens^121.
#cause^118@ag:==:#opacity^119.
#cause^118@th:==:#cataract.
#cause^125@th:==:#blindness.

>> Answer = #hepatitis

#disease@sub:==:#hepatitis.
#inflame^3@exp:==:#liver.
#inflame^3@ag:==:#hepatitis.
#cause^6@ag:==:#agents^7.
#agents^7@type_of:==:#infectious.
#agents^7@kind_of:==:#toxic.
#cause^6@th:==:#hepatitis.
#characterize^12@ag:==:[#jaundice,',',#fever,and,#enlargement^17].
#enlargement^17@body_part:==:#liver.
#characterize^12@th:==:#hepatitis.
#hepatitis@sub:==:#'Hepatitis A'.
#hepatitis@sub:==:#'Hepatitis B'.

>> Answer = #'Hepatitis A'

#hepatitis@sub:==:#'Hepatitis A'.
#cause^24@ag:==:#virus^25.
#persist^26@neg:==:y.
#persist^26@th:==:#virus^25.
#persist^26@loc_int:==:#serum^27.
#serum^27@of:==:#blood.
#cause^24@th:==:#'Hepatitis A'.
#'Hepatitis A'@duration:==:#acute.
#'Hepatitis A'@benignity:==:#benign.
#'Hepatitis A'@kind_of:==:#viral.

>> Answer = #'Hepatitis B'

#hepatitis@sub:==:#'Hepatitis B'.
#cause^32@ag:==:#virus^33.
#persist^35@th:==:#virus^33.
#persist^35@loc_int:==:#serum^27.
#serum^27@of:==:#blood.
#cause^32@th:==:#'Hepatitis B'.
#'Hepatitis B'@duration:==:#acute.
#'Hepatitis B'@kind_of:==:#viral.

>> Answer = #acne
```

```
#disease@sub:==:#acne.
#characterize^41@ag:==:#pimples^42.
#appear^43@mod:==:can.
#appear^43@th:==:#pimples^42.
#appear^43@loc_ext:==:#part^44.
#part^44@of:==:#body.
#characterize^41@th:==:#acne.
#cause^48@ag:==:#glands^49.
#glands^49@intensity:==:#overactive.
#glands^49@kind_of:==:#oil.
#cause^48@th:==:#acne.
#involve^54@perc:==:#glands^55.
#glands^55@kind_of:==:#sebaceous.
#glands^55@of:==:#skin.
#involve^54@exp:==:#acne.
#characterize^59@ag:==:[#papules,or,#pustules,or,#comedones].
#characterize^59@th:==:#acne.
#characterize^67@ag:==:[#papules^69,and,#pustules^70].
[#papules^69,and,#pustules^70]@loc_ext:==:[#face,and,#neck].
#characterize^67@th:==:#acne.
#acne@kind_of:==:#inflammatory.
#acne@body_part:==:#skin.

>> Answer = #'Addison\'s disease'

#disease@sub:==:#'Addison\'s disease'.
#disorder@sub:==:#'Addison\'s disease'.
#cause^78@ag:==:#deficiency^79.
#deficiency^79@loc_int:==:#hormones^80.
#hormones^80@kind_of:==:#adrenocortical.
#cause^78@th:==:#'Addison\'s disease'.
#characterize^83@ag:==:[#pigmentation^85,and,#pressure^88].
#pigmentation^85@color:==:#'bronze-like'.
#pigmentation^85@of:==:#skin.
#pressure^88@level:==:#low.
#pressure^88@of:==:#blood.
#characterize^83@th:==:#'Addison\'s disease'.
#cause^93@ag:==:#deficiency^94.
#deficiency^94@of:==:#hydrocortisone.
#cause^93@th:==:#'Addison\'s disease'.
#characterize^97@ag:==:[#anemia,',',#loss^102,and,#weakness^103].
#loss^102@of:==:#weight.
#weakness^103@degree:==:#extreme.
#characterize^97@th:==:#'Addison\'s disease'.
#'Addison\'s disease'@severity:==:#degenerative.
#'Addison\'s disease'@commonality:==:#rare.

>> Answer = #cataract
```

```
#disease@sub:==:#cataract.
#characterize^108@ag:==:#opacity^109.
#opacity^109@degree:==:#partial.
#opacity^109@completeness:==:#complete.
#opacity^109@loc_ext:==:[#lens,or,#capsule].
#opacity^109@loc_int:==:[#lens,or,#capsule].
#characterize^108@th:==:#cataract.
#cause^118@ag:==:#opacity^119.
#lens^121@body_part:==:#eye.
#opacity^119@of:==:#lens^121.
#impair^122@ag:==:#opacity^119.
#impair^122@exp:==:#vision.
#cause^118@th:==:#cataract.
#cataract@body_part:==:#eye.

>> Answer = #'Acute otitis media'

#disorder@sub:==:#'Acute otitis media'.
#involve^130@perc:==:[#inflammation^132,and,#infection].
[#inflammation^132,and,#infection]@of:==:#structures^134.
#structures^134@of:==:#ear^135.
#ear^135@position:==:#middle.
#involve^130@exp:==:#'Acute otitis media'.
#characterize^138@ag:==:#inflammation^139.
#inflammation^139@of:==:#ear^135.
#characterize^138@concomitant:==:#signs^142.
#signs^142@of:==:#infection.
#characterize^138@th:==:#'Acute otitis media'.
#cause^145@ag:==:#infection.
#cause^145@th:==:#'Acute otitis media'.

>> Question= [what,causes,something]
Number of syntagmas (OntoSeR-) from robust parser = 3
Number of syntagmas (OntoSeR+) validated by phi_onto = 1
OKR-query
#cause@ag:#:#what.
#cause@th:#:#something.

>> Answer = #agents^7

#cause^6@ag:==:#agents^7.
#agents^7@type_of:==:#infectious.
#agents^7@kind_of:==:#toxic.
#cause^6@th:==:#hepatitis.

>> Answer = #virus^25

#cause^24@ag:==:#virus^25.
```

```
#persist^26@neg:==:y.
#persist^26@th:==:#virus^25.
#persist^26@loc_int:==:#serum^27.
#serum^27@of:==:#blood.
#cause^24@th:==:#'Hepatitis A'.
#persist^35@th:==:#virus^33.
#persist^35@loc_int:==:#serum^27.

>> Answer = #virus^33

#cause^32@ag:==:#virus^33.
#persist^35@th:==:#virus^33.
#persist^35@loc_int:==:#serum^27.
#serum^27@of:==:#blood.
#cause^32@th:==:#'Hepatitis B'.
#persist^26@neg:==:y.
#persist^26@th:==:#virus^25.
#persist^26@loc_int:==:#serum^27.

>> Answer = #glands^49

#cause^48@ag:==:#glands^49.
#glands^49@intensity:==:#overactive.
#glands^49@kind_of:==:#oil.
#cause^48@th:==:#acne.

>> Answer = #deficiency^79

#cause^78@ag:==:#deficiency^79.
#deficiency^79@loc_int:==:#hormones^80.
#hormones^80@kind_of:==:#adrenocortical.
#cause^78@th:==:#'Addison\'s disease'.

>> Answer = #deficiency^94

#cause^93@ag:==:#deficiency^94.
#deficiency^94@of:==:#hydrocortisone.
#cause^93@th:==:#'Addison\'s disease'.

>> Answer = #opacity^119

#cause^118@ag:==:#opacity^119.
#lens^121@body_part:==:#eye.
#opacity^119@of:==:#lens^121.
#impair^122@ag:==:#opacity^119.
#impair^122@exp:==:#vision.
#cause^118@th:==:#cataract.
#cause^125@ag:==:#opacity^119.
#cause^125@th:==:#blindness.
```

```
>> Answer = #infection

#concept@sub:==:#infection.
#involve^130@perc:==:[#inflammation^132,and,#infection].
[#inflammation^132,and,#infection]@of:==:#structures^134.
#structures^134@of:==:#ear^135.
#ear^135@position:==:#middle.
#involve^130@exp:==:#'Acute otitis media'.
#signs^142@of:==:#infection.
#cause^145@ag:==:#infection.
#cause^145@th:==:#'Acute otitis media'.
#infection@kind_of:==:#bacterial.
#infection@kind_of:==:#viral.
#infection@of:==:#ear^135.
========================================================================
```

Appendix D

DEFINDER Evaluation

We have developed DEFINDER in the context of a medical digital library project, PERSIVAL, in order to automatically extract definitions from consumer-oriented medical articles (Klavans and Muresan, 2001; Muresan and Klavans, 2002). We thoroughly evaluated the system, on several dimensions: performance of the definition extraction algorithm in terms of precision and recall; quality of the generated dictionary as judged both by non-specialists and by medical specialists; coverage of on-line dictionaries (Klavans and Muresan, 2001; Muresan and Klavans, 2002).

D.1 Quantitative Evaluation

A standard approach in any system evaluation is to compare the results against human performance. Thus we selected four subjects, not trained in the medical domain and who did not participate in the development of the system. Each of them was provided with a set of nine articles, and was asked to annotate the definitions and their headwords in text. We equally represent the sources of our corpora (medical articles, book chapters, manuals and newspapers), but we limit the length of the articles to two pages.

The gold-standard against which we compared our system was determined by the set of definitions marked up by at least 3 out of the 4 subjects and consists of 53 definitions. Our system obtained 86.95% precision and 75.47% recall.

The interpretation of the results was more difficult then expected, given that there was no agreement among users regarding *what is a definition?*, even though they were provided with a set of instructions and sample definitions. For example, given the input sentence:

> The most frequent cause of the condition in older patients is atherosclerosis - the progressive narrowing of the heart's own arteries by cholesterol plaque buildups, which starves the heart itself for oxygen and nutrients.

our system identified as definition for *atherosclerosis* the whole phrase *the progressive narrowing of the heart's own arteries by cholesterol plaque buildups, [which starves the heart itself for oxygen and nutrients]*, while only 2 out of 4 subjects marked up the relative clause given in square brackets.

D.2 Qualitative Evaluation: Lay User Perspective

Satisfying both the specialist and the layman with a single definition of a technical term will be hard to achieve. Thus, in our next evaluation, our aim was to compare the quality of our lay dictionary against existing specialized dictionaries from the perspective of non-specialist users.

We chose the UMLS Metathesaurus and the On-line Medical Dictionary

Hypothesis	Usefulness	Readability
DEFINDER > UMLS	$p < 0.00003$	$p < 0.00003$
DEFINDER > OMD	$p < 0.00003$	$p < 0.00005$

Table D.1: Sign test (p) for usefulness and readability

(OMD)[1] as technical dictionaries. A set of eight subjects was provided with a list of randomly chosen 15 medical terms and their definitions from UMLS, OMD and the definition extracted by our system from on-line lay text. The source of each definition was not given in order not to bias the experiment. Also the order of definitions was randomly changed for each term. The task was to assign to each definition a quality rate (QR) for usefulness and readability on a scale of 1 to 7 (1 very poor, 7 excellent). Usefulness means that the definition can help the user understand the term, while readability means that the definition is not technical, thus is easy to read.

We first measured the Average Quality Rating (AQR) for each definition source on these two criteria. Our hypotheses were that DEFINDER outperforms both UMLS and OMD in terms of usefulness and readability. For usefulness, our system was rated 5.17, while OMD and UMLS obtained 3.9 and 2.94. In terms of readability, the difference was even higher: 5.65 compared with 4.3 and 3.18. In order to statistically validate our results we applied the sign test (Siegel and Castellan, 1988). As shown in Table D.1 by the **p** values, the results were statistically significant.

One question that arises in computing the AQR is whether the high scores given by one subject can compensate for the lower values given by other subject,

[1]http://www.graylab.ac.uk/omd

thus introducing noise in comparing the results. To address this issue we performed a second analysis to evaluate the relative ranking of the three definitional sources. Using Kendall's coefficient of correlation, W, we first measured the interjudge agreement on each term, and for terms with significant agreement we compute the level of correlation between them. If W was significant, we compared the overall mean ranks of the three sources. We tested the same hypotheses: DEFINDER is ranked better than UMLS and OMD both in terms of usefulness and readability. For usefulness, DEFINDER ranked first (1.3), OMD second (2.11) and UMLS third (2.57). For readability, we obtained the same relative ranking with the scores 1.25, 2.1 and 2.64, respectively. We obtained statistically significant W values (W=0.54 and W=0.45 at p=0.01 and p=0.05 respectively).

D.3 Qualitative Evaluation: Medical Specialist Perspective

The results of the previous section show that the definitions extracted from consumer-oriented text are readable and useful for the lay user, outperforming the existing specialized dictionaries. One question that arises is if they are also accurate and complete from medical point of view.

In order to answer this question we performed a user-based evaluation. We selected a set of 15 medical specialists (physician assistants, nurse practitioners, residents and medical students). Each subject was provided with the same set of 15 medical terms and the definitions extracted by DEFINDER from text, as the one given in the previous section. They were asked to judge the accuracy and completeness of the definitions on a scale from 1 to 7 (1 very poor, 7 excellent).

The definitions were rated on average 5.87 for accuracy and 5.38 for completeness. The results show that consumer-oriented text, when of high quality can be a valuable source of definitions. Also because our definitions were embedded in text, one of their required characteristics was to be concise. This explains the somewhat lower value obtained for completeness.

D.4 Coverage of Existing Dictionaries

In this study we evaluated the coverage of three on-line dictionaries. In the introduction we claimed that these dictionaries are incomplete and our system can be used to fill in the gaps. We selected two specialized dictionaries: UMLS Metathesaurus and On-line Medical Dictionary, and one popular glossary: Glossary of Popular and Technical Medical Terms (GPTMT).[2] The popular glossary was chosen since it would be a good resource for lay users and we wanted to analyze its completeness. A base test set of 93 terms and their associated definitions was chosen for this experiment. As expected three cases were found:

1. the term is listed in one of the on-line dictionaries and is defined in that dictionary (*defined*)

2. the term is listed in one of the on-line dictionaries but does not have an associated definition (*undefined*)

3. the term is not listed in one of the on-line dictionaries (*absent*)

Results are presented in Table D.2. Looking at the UMLS results, we noticed that 24% of terms were undefined, which is equivalent to say that they are in the

[2]http://allserv.rug.ac.be/%7Ervdstich/eugloss/welcome.html

Term	UMLS	OMD	GPTMT
defined	60%(56)	76%(71)	21.5%(20)
undefined	24%(22)	-	-
absent	16%(15)	24%(22)	78.5%(73)

Table D.2: Coverage of on-line dictionaries

axiomatic vocabulary. But the question is if these terms are really known by the lay users (e.g., *Holter monitor* or *coumadin*)? Analyzing the terms that were classified as absent in UMLS, we conclude that modifiers play an important role in deciding which are the true terms (e.g. *cardiac defibrillator* was the defined term extracted by our system, while in UMLS only the term *defibrillator* was present).

In the case of the popular dictionary (GPTMT) only 20 out of the 93 terms were present, thus achieving a coverage of only 21.5%. These results encourage us to believe that building dictionaries automatically from text is a valuable endeavor for enhancing existing resources.

References

Adriaans, Pieter and Erik Haas. 1999. Grammar induction as substructural inductive logic programming. In James Cussens, editor, *Proceedings of the 1st Workshop on Learning Language in Logic*, pages 117–127, Bled, Slovenia.

Anderson, Stephen R. 2004. Some points of agreement. In *Johns Hopkins IGERT Workshop.*

Angluin, Dana. 1982. Inference of reversible languages. *Journal of the Association for Computing Machinery*, 29(3):741–765.

Avgustinova, Tania and Hans Uszkoreit. 2001. Towards a typology of agreement phenomena. In W. Griffin, editor, *The Role of Agreement in Natural Languages. Texas Linguistic Society Conference (TLS'2001)*, Austin, Texas, USA. Lincom Europa.

Baker, Collin, Charles Fillmore, and John Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING/ACL '98)*, pages 86–90, Montreal, Canada.

Bateman, John A. 1992. The theoretical status of ontologies in natural language processing. In S. Preuß and B. Schmitz, editors, *Text Representation and Domain Modeling – ideas from linguistics and AI.* Technische Universitat Berlin, pages 50 – 99.

Blackburn, Patrick and Johan Bos. 2005. *Representation and Inference for Natural Language. A First Course in Computational Semantics.* CSLI Publications.

Brazma, Alviz. 1993. Efficient identification of regular expressions from representative examples. In *Proceedings of the Computational Learning Theory Conference*, pages 236–242.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Oxford: Blackwell.

Brill, Eric. 1992. A simple rule-based part of speech tagger. In *Proceedings of the 3rd Applied Natural Language Processing Conference (ANLP'92)*, Trento, Italy.

Carreras, Xavier and Lluis Marquez. 2004. Introduction to CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of The Eight Conference on Natural Language Learning*.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics (NAACL-2000)*, Seattle, Washington.

Chaudhri, Vinay K., Adam Farquhar, Richard Fikes, Peter D. Karp, and James Rice. 1998. OKBC: A programmatic foundation for knowledge base interoperability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, Wisconsin.

Chen, John and Owen Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*, Sapporo, Japan.

Chodorow, Martin S., Roy J. Byrd, and George Heidorn. 1985. Extracting semantic hierarchies from a large on-line dictionary. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 299–304, Chicago.

Chomsky, Noam. 1970. Remarks on nominalization. In R. Kacobs and P. Rosenbaum, editors, *Readings in English Transformational Grammar*. Blaisdell, Waltham, MA.

Clark, Alexander. 2001. *Unsupervised Language Acquisition: Theory and Practice*. Ph.D. thesis, University of Sussex.

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

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

Copestake, Ann and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage english grammar using HPSG. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*.

Copestake, Ann, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2001. Minimal Recursion Semantics: An introduction. *Language and Computation*, 1(3).

Copestake, Ann, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 1999. Minimal Recursion Semantics: An introduction.

Cussens, James and Sao Džeroski, editors. 2000. *Learning Language in Logic, volume 1925 of Lecture Notes in Computer Science*. Springer-Verlag.

De Raedt, Luc, editor. 1996. *Advances in Inductive Logic Programming*. IOS Press, Amsterdam, The Netherlands.

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

Diessel, Holger and Michael Tomasello. 2005. A new look at the acquisition of relative clauses. *Language*, 81:1–25.

Dorr, Bonnie J. 1997. Large-scale dictionary construction for foreign language tutoring and interlingual machine translation. *Machine Translation*, 12(4):271–322.

Dupont, Pierre, Laurent Miclet, and Enrique Vidal. 1994. What is the search space of the regular inference? In R. C. Carrasco and J. Oncina, editors, *Proceedings of the Second International Colloquium on Grammatical Inference (ICGI-94)*, volume 862, pages 25–37, Berlin. Springer.

Džeroski, Sašo and Nada Lavrač, editors. 2001. *Relational Data Mining*. Springer, Berlin, Germany.

Eck, Karen E. and Ingrid E. Meyer. 1995. Bringing Aristotle into the 20th century: Computer-assisted definition construction in a terminological knowledge base. In Sue Ellen Wright and Richard A. Strehlow, editors, *Standard-*

izing and Harmonizing Terminology: Theory and Practice. ASTM, Philadelphia, pages 83–101.

Flickinger, Dan and Emily Bender. 2003. Compositional semantics in a multilingual grammar resource. In *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*.

Fodor, Janet Dean. 1978. Parsing strategies and constraints on transformations. *Linguistic Inquiry*, 9:427–473.

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

Gazdar, Gerald, Ewan Klein Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985. *Generalized Phrase Structure Grammars*. Blackwell Publishing, Oxford and Harvard University Press, Cambridge, MA.

Gierz, Gerhard, Karl Heinrich Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael Mislove, and Dana S. Scott. 2003. *Continuous Lattices and Domains*, volume 93 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press.

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

Gold, E. M. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

Grishman, Ralph, Catherine Macleod, and Adam Meyers. 1994. COMLEX syntax: Building a computational lexicon. In *Proceeding of 15th International Conference on Computational Linguistics (COLING-1994)*, Kyoto, Japan.

Groenendijk, Jeroen and Martin Stokhof. 1984. On the semantics of questions and the pragmatics of answers. In F. Landman and F. Veltman, editors, *Varieties of Formal Semantics.*

Gruber, Thomas R. 1993. A translational approach to portable ontology specification. *Knowledge Acquisition*, 5:199–220.

Harper, Mary P., Wen Wang, and Christopher M. White. 2001. Approaches for learning constraint dependency grammar from corpora. In *Proceedings of the Grammar and Natural Language Processing Conference*, Montreal, Canada.

Hirst, Graeme. 1987. *Semantic interpretation and the resolution of ambiguity.* Cambridge University Press, New York, NY, USA.

Hirst, Graeme. 2003. Ontology and the lexicon. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies in Information Systems.* Berlin: Springer.

Hobbs, Jerry R. 1985. Ontological promiscuity. In *Proceedings of the 25th Annual Association for Computational Linguistics (ACL-85)*, pages 61–69.

Hwa, Rebecca. 1999. Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99).*

Jackendoff, Ray. 1977. *X-Syntax: a Study of Phrase Structure.* The MIT Press:Cambridge Mass.

Janssen, Theo M. V. 1997. Compositionality. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language,and Linguistics*. Elsevier Science, pages 417–473.

Jensen, Per Anker and Jorgen Fischer Nilsson. 2003. Ontology-based semantics of prepositions. In *Proceedings of ACL-SIGSEM Workshop: The Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*.

Johannessen, Janne Bondi. 1997. *Coordination*. Oxford University Press.

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

Kaplan, Ronald and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*. Cambridge, MA: The MIT Press, pages 173–281.

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

Kay, Martin. 1979. Functional grammar. In *Proceedings of the Fifth Meeting of the Berkeley Linguistics Society*, pages 142–158, Berkeley, CA.

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

Klavans, Judith, Martin Chodorow, and Nina Wacholder. 1992. Building a knowledge base from parsed definitions. In George Heidorn, Karen Jensen, and Steve Richardson, editors, *Natural Language Processing: The PLNLP Approach*. Kluwer, New York, pages 119–133.

Klavans, Judith and Smaranda Muresan. 2000. DEFINDER: Rule-Based Methods for the Extraction of Medical Terminology and their Associated Definitions from On-line Text. In *Proceedings of the American Medical Informatics Association Symposium (AMIA-2000)*.

Klavans, Judith and Smaranda Muresan. 2001. Evaluation of DEFINDER: A System to Mine Definitions from Consumer-oriented Medical Text. In *Proceedings of The First ACM+IEEE Joint Conference on Digital Libraries*.

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

Kowalski, Robert .A. 1979. *Logic for Problem Solving*. North-Holland Publishing Co., Amsterdam, The Netherlands.

Labelle, Marie. 1990. Prediction, wh-movement and the development of relative clauses. *Language Acquisition*, 1:95–120.

Lange, Steffen, Jochen Nessel, and Rolf Wiehage. 1998. Learning recursive languages from good examples. *Annals of Mathematics and Artificial Intelligence*, 23(1-2):27–52.

Lauer, Mark. 1995. *Designing Statistical Language Learner: Experiments on Noun Compounds*. Ph.D. thesis, Sydney University.

Lindberg, Donald, Betsy Humphreys, and Alexa McCray. 1993. The unified medical language system. *Methods of Information in Medicine*, 32:281–291.

Lloyd, John W. 2003. *Logic for Learning: Learning Comprehensible Theories from Structured Data*. Springer, Cognitive Technologies Series.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

McDonald, David D. 2001. The view from the trenches: issues in the ontology of restricted domains. In *International Conference on Formal Ontology in Information Systems (FOIS-2001)*, pages 22–33.

Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank Project: An interim report. In *Proceedings of HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31.

Miller, George. 1990. WordNet: An on-line lexical database. *Journal of Lexicography*, 3(4):235–312.

Miller, Scott, Robert Bobrow, Robert Ingria, and Richard Schwartz. 1994. Hidden understanding models of natural language. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL'94)*, pages 25–32.

Miltsakaki, Eleni, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The Penn Discourse TreeBank. In *Language Resources and Evaluation Conference (LREC-04)*.

Moldovan, Dan I. and Vasile Rus. 2001. Logic form transformation of WordNet and its applicability to question answering. In *Proceedings of the 39th Annual Meeting and 10th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL 2001)*, pages 394–401.

Montague, Richard. 1973. The proper treatment of quantification in ordinary english. In K.J.J Hintikka, J.M.E. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language*. Dordrecht: Reidel, pages 221–242.

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

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

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

Muresan, Smaranda. 2005. Parsing preserving techniques in grammar induction. Technical Report CUCS-032-05, Columbia University, New York, NY.

Muresan, Smaranda and Judith L. Klavans. 2002. A method for automatically building and evaluating dictionary resources. In *Proceedings of the Language Resources and Evaluation Conference (LREC-2002)*.

Muresan, Smaranda, Tudor Muresan, and Judith Klavans. 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, Smaranda, Tudor Muresan, and Judith Klavans. 2005. Lexicalized Well-Founded Grammars: Learnability and merging. Technical Report CUCS-027-05, Columbia University, New York, NY.

Muresan, Smaranda, Tudor Muresan, and Rodica Potolea. 2002. Data flow coherence constraints for pruning the search space in ILP tools. *International Journal of Artificial Intelligence Tools*, 11(2).

Muresan, Tudor, Rodica Potolea, and Smaranda Muresan. 1998. Amalgamating CCP with Prolog. *Scientific Journal of Polytechnic University, Timisoara*, 43(4).

Neumann, Günter and Gertjan van Noord. 1994. Reversibility and self-monitoring in natural language generation. In Tomek Strzalkowski, editor, *Reversible Grammar in Natural Language Processing*. Kluwer Academic Publishers, Boston, pages 59–96.

Nirenburg, Sergei and Victor Raskin. 2004. *Ontological Semantics*. MIT Press.

Norman, Guy. 2002. Description and prescription in dictionaries of scientific terms. *International Journal on Lexicography*, 15(4).

Osborne, Miles. 1999. DCG induction using MDL and parsed corpora. In James

Cussens, editor, *Proceedings of the 1st Workshop on Learning Language in Logic*, pages 63–71, Bled, Slovenia.

Palmer, Martha, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1).

Pereira, Fernando C. and Stuart M. Shieber. 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*, pages 123–129.

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

Pereira, Fernando C. and David H.D Warren. 1983. Parsing as deduction. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 137–144.

Philips, Colin, Nina Kazanina, and Shani H. Abada. 2005. ERP effects of the processing of syntactic long-distance dependencies. *Cognitive Brain Research*, 22(3):407–428.

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

Poesio, Massimo and Renata Vieira. 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2).

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

Proietti, Maurizio and Alberto Pettorossi. 1999. Transforming inductive definitions. In *The 16th International Conference on Logic Programming*, pages 486–499.

Pustejovsky, James. 1995. *The Generative Lexicon.* MIT Press.

Pustejovsky, James, Adam Meyers, Martha Palmer, and Massimo Poesio. 2005. Merging PropBank, NomBank, TimeBank, Penn Discourse Treebank and Coreference. In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*, pages 5–12, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Quirk, Randolph, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1972. *A Grammar of Contemporary English.* Longman.

Ramshaw, Lance A. and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of Third ACL Workshop on Very Large Corpora.*

Richardson, Stephen, William Dollan, and Lucy Vanderwende. 1998. MindNet: Acquiring and structuring semantic information from text. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING/ACL '98).*

Rosario, Barbara and Marti Hearst. 2001. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *Proceedings of*

the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-2001), Pittsburgh, PA.

Rus, Vasile. 2002. *Logic Form For WordNet Glosses and Application to Question Answering.* Ph.D. thesis, Computer Science Department, School of Engineering, Southern Methodist University, Dallas, Texas.

Sag, Ivan A. 1997. English relative clause constructions. *Journal of Linguistics*, 33:431–484.

Sag, Ivan A. and Thomas Wasow. 1999. *Syntactic Theory: A formal introduction.* Stanford: CSLI Publications.

Sag, Ivan A., Thomas Wasow, and Emily Bender. 2003. *Syntactic Theory: A formal introduction, Second Edition.* Stanford: CSLI Publications.

Saint-Dizier, Patrick. 2005. An overview of PrepNet: Abstract notions, frames and inferential patterns. In *The Second ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, Colchester, United Kingdom.

Sakakibara, Yasubumi. 1992. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97(1):23–60.

Sakakibara, Yasubumi. 1997. Recent advances of grammatical inference. *Theoretical Computer Science*, 185(1):15–45.

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

Seki, Shinnosuke and Satoshi Kobayashi. 2004. Efficient learning of k-reversible context-free grammars from positive structural examples. In *Proceedings of the 7th International Colloquium of Grammatical Inference (ICGI-2004)*, pages 285–287.

Shieber, Stuart. 1994. The problem of logical-form equivalence. *Computational Linguistics*, 19(1):179–190.

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

Shieber, Stuart, 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*. SRI International, Menlo Park, CA, November, techreport 4, pages 39–79. Final report for SRI Project 1894.

Shieber, Stuart M. 1986. A simple reconstruction of GPSG. In *Proceedings of the 11th International Conference on Computational Linguistics*, pages 211–216, Bonn, West Germany.

Shieber, Stuart M. 1992. *Constraint-Based Grammar Formalisms*. MIT Press.

Siegel, Sidney and John N. Castellan. 1988. *Nonparametric statistics for the behavioral sciences*. McGraw Hill, Boston, MA.

Sowa, John F. 1999. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA.

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

Steels, Luc. 2004. Constructivist development of grounded construction grammars. In *Proceedings of the Annual Meeting for the Association for Computational Linguistics Conference (ACL-2004)*, Barcelona, Spain.

Steels, Luc, Joachim De Beule, and Nicolas Neubauer. 2005. Linking in fluid construction grammar. In *Proceedings of The 17th Belgian-Dutch Conference on Artificial Intelligence (BNAIC-05)*, Brussels, Belgium. Royal Flemish Academy for Science and Art.

Tang, Lappoon R. and Raymond Mooney. 2000. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In *The Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 133 – 141.

Tang, Lappoon R. and Raymond Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *The 12th European Conference on Machine Learning (ECML-2001)*, pages 466–477.

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

Valiant, Leslie G. 1984. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

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

van Noord, Gertjan. 1993. *Reversibility in Natural Language Processing.* Ph.D. thesis, University of Utrecht.

Voorhees, Ellen M. 1999. The trec-8 question answering track report. In *Proceeding of the Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, Maryland. National Institute of Standards and Technology (NIST).

Weisman, Herman M. 1992. *Basic Technical Writing.* New York: Merrill.

Wilks, Yorick A., Brian M. Slator, and Louise Guthrie. 1996. *Electric Words: Dictionaries, Computers, and Meanings.* MIT Press.

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

Wintner, Shuly. 2002. Modular context-free grammars. *Grammars*, 5:41–63.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, Barcelona, Spain.

Zelle, John M. and Raymond. J. Mooney. 1993. Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 817–822, Washington, D.C. AAAI Press/MIT Press.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 658–66, Arlington, Virginia. AUAI Press.