Chapter 1

Where do we stand?

This course is part of the "Logic & Language" Session of ESSLLI'04. As such, we would like to start in this preliminary and informal chapter from the very broad area and zoom into the framework we will be introducing in the course. In this way we hope to help the reader grasping the main properties of the framework and spotting similarities and differences with what he/she labels as "Logic & Language". We will start from Computational Linguistics, move to Formal Grammars and end with Logical Grammars.

1.1 Formal Grammars

Formal Grammars are an area of investigation of *Computational Linguistics*. The field is as diverse as linguistics itself, in general it could been seen as the study of *formal devices* which can be used to distinguish grammatical and ungrammatical strings of a given language [CCCSS00] and build their semantic interpretation. These formal devices can be automata, which are abstract computing machines, and formal grammars, which are string (or structure) rewriting systems. Formal Grammars are used by parsers to determine the "syntactic structure" of string of words. See [Mit03] for an overview of the field.

The object of this course is a formal grammar. Here we point out the major tasks of formal grammars and the devices that are used to carry them out by briefly comparing Context-Free Phrase Structure Grammars with the framework object of our studying, Categorial Grammars (CG)[1].

Phrase Structure Grammars. In formal language theory a language is defined as a set of strings, i.e. a set of finite sequences of vocabulary items. A grammar for a language then is a formal device that defines which strings belong to that language. One particular kind of formal system that is used to define a language is commonly known as a *context-free phrase structure grammar*. Besides deciding whether a string

[1]This section is extracted (and adapted) from [Hey99] with the author's permission. We thank Dirk for this.

belongs to a given language, this grammar deals with phrase structures represented as trees.

An important difference between strings and phrase structures is that whereas string concatenation is assumed to be associative, trees are bracketed structures. The latter thus preserve aspects of the compositional (constituent) structure or derivation which is lost in the string representations. We illustrate the behavior of this grammar by means of the example below.

EXAMPLE 1.1. [Grammars and Languages]We consider a small fragment of English defined by the following grammar G ($= \langle \mathsf{LEX}, \mathsf{RULES} \rangle$), with vocabulary Σ and categories CAT.

$\Sigma = \{\text{Sara}, \text{dress}, \text{wears}, \text{the}, \text{new}\}$,
$\mathsf{CAT} = \{det, n, np, s, v, vp, adj\}$,
$\mathsf{LEX} = \{\langle \text{Sara}, np\rangle, \langle \text{the}, det\rangle, \langle \text{dress}, n\rangle, \langle \text{new}, adj\rangle, \langle \text{wears}, v\rangle\}$
$\mathsf{RULES} = \{s \rightarrow np\ vp,\ np \rightarrow det\ n,\ vp \rightarrow v\ np, n \rightarrow adj\ n\}$

Among the elements of the language recognized by the grammar, $L(G)$, are $\langle$the, $det\rangle$ because this is in the lexicon, and $\langle$Sara wears the new dress, $s\rangle$ which is in the language by the repeated application rules.

(1) $\langle$new dress, $n\rangle \in L(G)$ because
$n \rightarrow adj\ n \in \mathsf{RULES}$,
$\langle$new, $adj\rangle \in L(G)$ (LEX), and
$\langle$dress, $n\rangle \in L(G)$ (LEX)

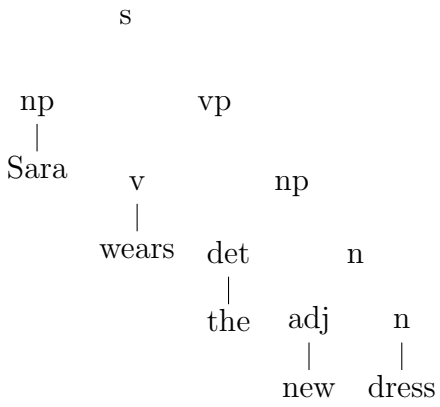
(2) $\langle$the new dress, $np\rangle \in L(G)$ because
$np \rightarrow det\ n \in \mathsf{RULES}$,
$\langle$the, $det\rangle \in L(G)$ (LEX), and
$\langle$new dress, $n\rangle \in L(G)$ (1)

(3) $\langle$wears the new dress, $vp\rangle \in L(G)$ because
$vp \rightarrow v\ np \in \mathsf{RULES}$,
$\langle$wears, $v\rangle \in L(G)$ (LEX), and
$\langle$the new dress, $np\rangle \in L(G)$ (2)

(4) $\langle$Sara wears the new dress, $s\rangle \in L(G)$ because
$s \rightarrow np\ vp \in \mathsf{RULES}$,
$\langle$Sara, $np\rangle \in L(G)$ (LEX), and
$\langle$wears the new dress, $vp\rangle \in L(G)$ (3)

In a similar way we can show that the set of phrase structure trees ($T(G)$) contains the following tree which we also depict in the usual graphical way.

$\langle\langle \text{Sara}, np\rangle, \langle\langle \text{wears}, v\rangle, \langle\langle \text{the}, det\rangle, \langle\langle \text{new}, adj\rangle, \langle \text{dress}, n\rangle, n\rangle, np\rangle, vp\rangle, s\rangle$



As the following table shows, we can also derive that the sentence is in the language by means of a rewriting procedure. The left column represents the sequence of categories and words that is arrived at by replacing one of the categories (c) (identical to the left-hand side of the rule in the second column) on the line above by the right-hand side of the rule or by a word that is assigned the category c by the lexicon.

s	$s \rightarrow np\ vp$
np vp	$vp \rightarrow v\ np$
np v np	$np \rightarrow det\ n$
np v det n	$n \rightarrow adj\ n$
np v det adj n	$\langle \text{Sara}, np \rangle$
Sara v det adj n	$\langle \text{wears}, v \rangle$
Sara wears det adj n	$\langle \text{the}, det \rangle$
Sara wears the det adj n	$\langle \text{new}, adj \rangle$
Sara wears the new n	$\langle \text{dress}, n \rangle$
Sara wears the new dress	

This example recapitulates the basic definitions of grammar and language isolating some of the functions a grammar is required to serve. In particular, it shows that the principal tasks of a grammar are the definition of several sets of objects: (i) a set of expressions (the string set), (ii) a set of pairs of expressions and categories (language), (iii) a set of phrase structures. More abstractly, we can say that a grammar is a device that is concerned with two aspects.

(1) Defining the membership of elements to some (sub)set: *classification* or categorisation.

(2) Specifying the *compositional* structure of complex elements.

The grammars that we introduced categorise expressions in two steps. The lexicon component LEX of a grammar deals with the categorisation of the atomic expressions (the words), whereas the RULES determine the category of complex expressions. The

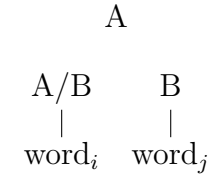
rules also take care of the definition of compositional structure. They define which parts can combine to form larger structures. Categories play an important role in defining the compositional structure because they group together the set of expressions that behave similarly with respect to compositional structure. In other words, the rules make reference to the categories, not to the individual expressions.

Besides classification, another important function of a grammar is to define structure. Analysing this aspect, we can say that the context-free grammar fixes the order of (sub)expressions (precedence) and a part-whole structure (dominance). Phrase structure trees make this compositional structure and the categorisation of wholes and parts explicit. We will now turn to another type of grammar that performs the same tasks in a different way and which is the ancestor of the logical grammar presented in the course.

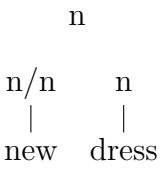
Categorial Grammars. The formal grammar we work with, Categorial Grammars (CG), adopts an alternative way to define classification and composition. In this type of grammar the lexical component takes over the role of the rule component in fixing the compositional structure of a language. It is important to notice how in order to effect this, the notion of category is modified.

CG have a richer notion of category than Phrase Structure Grammars. Besides atomic categories, they also use complex categories of the form A/B and $(B\backslash A)$ (where A and B are again categories, possibly complex themselves.) A language is defined solely by a lexicon that assigns categories to words. There are no phrase structure rules. Instead, a pair of general combination operations is used: (1) an expression in A/B concatenated with an expression of category B gives an expression in A (see (a) below); (2) an expression in B concatenated with an expression in $B\backslash A$ gives an expression in A (the schema is symmetric to (a)). Notice that these schemata do not mention specific categories, but use variables over arbitrary categories A, B. The information about precisely which expressions combine with others is provided by the lexicon. For instance, an adjective, such as *new*, might be assigned the category n/n, which, by the general schema, means that it combines with expressions in n to form expressions in n. Representing this composition in a tree we have (a') that plays the role of the phrase structure in (b).

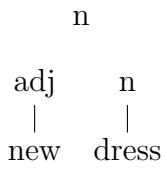
(a)



(a')



(b)



Note that CG categories could be seen as trees themselves and therefore are close to the elementary trees of Tree Adjoining Grammar (TAG). The reader is referred to the foundational course given by B. Gaiffe and G. Perrier "Basic Parsing Techniques for natural language" (ESSLLI 04) for an introductory comparison of CG and TAG and to [Moo02] for a formal embedding of (a version of) the former into the latter.

From this elementary summary, it should be clear how classification and composition are defined in this alternative framework. One thing that is important to underline is that the classification of expressions by the categories directly expresses their combination properties. These aspects will be spelled out in precise terms in Chapter 4 where it is also shown how moving from Classical CG to Categorial Type Logic (CTL) corresponds to move from a formal grammar to a logic grammar where categories are logical formula and rules are logical rules. Here we try to briefly explain the place CTL occupies within the "Logic and Natural Language Processing" (LNLP) area.

1.2 Logic and NLP

In several fields Logic is used to grasp the global features of reality abstracting away from the empirical details and providing a model of the observed objects. In our case, the object is natural language. Logic has been recognized to be an important tool for understanding the structure of language (both its syntax and semantics) since [Fre87]. Moreover, inference tools are a crucial component of Natural Language Processing (NLP) systems that require deep analysis which integrate semantic analysis of natural language utterances to their general cognitive processing. Valuable references for foundational and advanced introductions to the field are [PMW90, vBtM97].

Since 1995 an European Conference on *Logical Aspects of Computational Linguistics* (LACL) is organized every two years. In the introduction of LACL'96 proceedings the field is classified in (i) logical semantics of natural language, (ii) grammar and logic, (iii) mathematics with linguistic motivations, and (iv) computational perspectives. Following this division, if we have to chose one label, we can say that the framework presented in the course follows under the "grammar and logic" one. As before, we would like to highlight the main aspects in which CTL differs from other logical views on grammars.

Logic Based Grammars. One known example of a family of logics used to accomplish NLP tasks are Feature Logics [Rou97]. They are known to be especially useful in classifying and constraining the linguistic objects known as feature structures. Here, the logic is used to *formalize* linguistic theories. For instance, the feature co-occurrence restriction used in Generalized Phrase Structure Grammar (GPSG) to say that any category classified as a verb must have a negative "noun" and a positive "verb" quality

$$[VFORM] \supset [-N, +V]$$

can be represented in modal feature logic by means of the universal modality $\Box$ as following

$$\Box(vform : ture \rightarrow n : - \wedge v : +)$$

In other words, a linguistics rule is "translated" into a logical language and then the formal system is used to reason with the formalized linguistic objects.

Contrary to this, in CTL there is no such distinction between an a-priori grammar (with its set of grammatical rules) and a logic in which the former is translated. The grammar is in itself a logic, and the only rules at work are the (logical and structural)

rules of the system obtained by investigating its algebraic structure. The reader is invited to compare the use of the universal modality above with its application in CTL explained in Part II.

As fully explained in these course notes, CTL is a (modal) logic used to reason on linguistic signs. Syntactic categories are formulas, grammatical rules are logical rules, and structural (re)ordering corresponds to structural rules (or frame constraints). As such, *soundness* and *completeness* properties do not simply characterize the logic, but rather are main properties of the grammar itself. The challenge is to investigate the 'theoretical space' of the family of (modal) logics and select the proper model of natural language grammar suitable for the specific natural language to be generated. In the course, we will show how Display Calculi help carrying out this task.

Resources Sensitivity. A second crucial point is the *resource sensitivity* notion that governs the manner in which the operations of a system can utilize its resources: how often they may be used, how they can be assembled together to create larger structures, and how they can be reconfigured into other equivalent structures.

From a logical perspective, resource sensitivity is nicely exemplified in Linear Logic [Gir87] where the implicational connective $\multimap$ is used to indicate that it consumes the resource that is needed to prove the consequent; thus, the formula $p \multimap q$ is read as 'consume p yielding q'. After the rule for $\multimap$ is applied, the resource p is no longer available for further inferential steps.

In Linear Logic, Contraction and Multiplication of resources are reintroduced locally by means of unary operators $(!, ?)$ which are used to mark those resources for which the corrisponding structural rules are adequate. The use of these "markers" introduces the idea of structural rules that are controlled rather then globally available.

The logical concerns behind this logic have direct parallels in natural language grammar and are reflected in CTL which is a resource-sensitive logic (see [Moo99] for more details). Clearly, the multiplicity of linguistics material is important, since linguistic elements must generally be used once and only once during an analysis. Thus, we cannot ignore or waste linguistic material (1-a), nor can we indiscriminately duplicate it (1-b).

(1) a. *The coach smiled the ball. $\neq$ The coach smiled.

b. *The coach smiled smiled. $\neq$ The coach smiled.

As in other domains, in Linguistic as well, there is the need of locally controlling structural reasoning and account for the different compositional relations linguistic phenomena may exhibit. As in Linear Logic, in CTL as well this control is expressed by means of unary operators. However, the unary operators of CTL belong to the same algebraic structure of the implication operators used for simple composition of structures. This algebraic property and its encoding into sequent calculi is the topic of Chapter 3.

Statistical and Logic Based Approaches. We would like to conclude this part with a last comment on the possible application of logical grammars. As pointed out in [Car03], while, logic oriented linguists are more interested in selecting the proper logic

for modeling natural language, computer scientists typically worry about efficiency of processing in terms of time and space, often side-stepping cognitive issues in the interest of building effective software. Within the computational linguistic community, this interest is reflected in the development of statistical models that have largely supplanted logical ones. Statistical algorithms search for the "best" structure which is assumed to be the one with the highest likelihood and perform much better than any logic-based grammar system when put at work with large data, as needed in several applications. Rather than seeing this as a failure of the logical approach, in the LNLP area the integration of the inductive and the deductive styles of grammatical reasoning is perceived as an exciting challenge for the field. (See [Per00] for an interesting discussion.) As for the computational aspects of CTL and related grammars, the reader is refered to the courses given at ESSLLI'04 by P. Casteran and R. Moot (Proof automation for type-logical grammars) and by G. Jaeger and J. Michaelis (An introduction to mildly context-sensitive grammar).