

1. The Task of Grammar

Language-use consists in the processing [production and recognition] of signs/symbols. Signs are variously realized – by (1) gestures, (2) sounds, (3) graphs – which correspond respectively to (1) gesticular language,¹ (2) spoken language, (3) written language.

The signs of a language are produced by combining various smallest units – "signemes" – for example, gestemes, phonemes, graphemes. Some combinations of signemes are admissible, and others are inadmissible, in a given language.

The *principal* task of grammar is the demarcation of signeme-combinations of a given language into those that are admissible and those that are inadmissible. As originally proposed by Chomsky, grammar has three components, respectively known as *syntax*, *semantics*, and *phonology*. These three components of grammar are distinct inasmuch as they judge admissibility on the basis of different criteria. On the one hand, semantics judges admissibility with regard to *meaning*, and phonology judges admissibility with regard to *pronunciation*.² On the other hand, syntax judges admissibility without regard to either meaning or pronunciation, but by more abstract criteria that ultimately can only be described as "syntactic". Syntax is that part of grammar that remains after you remove meaning and pronunciation.

2. Syntax

We begin our investigation with syntax, since that is generally thought to be the core of grammar; indeed, for many, 'grammar' just means syntax. The idea is that, central to our language-processing facility is a syntactic-module, which delivers material to the other two language modules for further processing – the phonetic-module (pronunciation) and the semantic-module (understanding).

Since we are ultimately concerned with semantics, we will be primarily interested in the syntactic material that is delivered to the semantic-module, and we will largely ignore the material that is delivered to the phonetic-module. This allows us to examine considerably simpler syntactic models than we would have to otherwise.

3. Artificial versus Natural Languages

This brings us to the distinction between artificial and natural languages. An artificial language is created from scratch, and its grammar is a simple matter of proclamation. By contrast, a natural language is a social-historical phenomenon whose grammar must be discovered. Natural languages, like most realms of the natural world, are mind-bogglingly complex, and uncovering their grammar is an on-going enterprise, just like other branches of science.

In recent years, it has become fairly standard practice to use artificial-language tools to study natural-language grammars. This is accomplished by examining various well-defined fragments of natural language, and treating them (in effect) as artificial languages. The goal then is to produce ever-richer and more accurate fragments of the natural languages under scrutiny.

¹ Including "body" language and "sign" language.

² Narrowly understood, phonology pertains to pronunciation, and accordingly only applies to spoken languages – or "tongues" as they are sometimes called. However, we can also understand phonology more generally to pertain to whatever procedure is used to physically realize the symbols of a language.

4. Example 1 – Sentential Logic

One of the simplest examples of syntax occurs in elementary logic – the syntax of sentential logic (SL). Traditionally, formal languages are characterized by two features.

- (1) vocabulary
- (2) rules of formation

In the case of SL, this may be implemented as follows.

1. Vocabulary

1. upper case Roman letters (possibly with "decorations")
2. logical symbols: \sim , $\&$, \vee , \rightarrow , \leftrightarrow
3. punctuation marks: (,)
4. nothing else.

2. Rules of Formation

1. every upper-case Roman letter (decorated or undecorated) is a formula;
2. if \mathcal{A} is a formula, then so is: $\sim\mathcal{A}$;
if \mathcal{A} and \mathcal{B} are formulas, then so are: $(\mathcal{A}\&\mathcal{B})$, $(\mathcal{A}\vee\mathcal{B})$, $(\mathcal{A}\rightarrow\mathcal{B})$, $(\mathcal{A}\leftrightarrow\mathcal{B})$;
3. nothing else is a formula.

5. Example 2 – A Fragment of English

The following is a syntax for a very simple fragment of English.

- (1) a noun-phrase followed by a verb-phrase is a sentence
- (2) an attitude-verb followed by a sentence is a verb-phrase
- (3) 'believes' is an attitude-verb
- (4) 'Jay' is a noun-phrase
- (5) 'Kay' is a noun-phrase
- (6) 'is reading' is a verb-phrase

6. Category-Governed Grammars

The grammar (syntax) presented in the previous section is an example of a **category-governed** grammar. To say that a grammar \mathbb{G} is category-governed is to say the following.

- (1) every significant item of \mathbb{G} is assigned a *category*;
- (2) every construction rule of \mathbb{G} is *category-governed*.

In traditional syntax, categories very closely resemble the "parts of speech" that appear in dictionaries. For example, 'lie' is both a *verb* and a *noun*, the latter being very prominent categories in traditional grammar.

In this connection, the following is a relevant definition given at SIL.org.

A syntactic category is a set of words and/or phrases in a language which share a significant number of common characteristics. The classification is based on similar structure and sameness of distribution (the structural relationships between these elements and other items in a larger grammatical structure), and not on meaning. In generative grammar, a syntactic category is symbolized by a node label in a constituent structure tree.

For example, in Example 2, the categories are the following.

- (1) sentence
- (2) noun-phrase
- (3) verb-phrase
- (4) attitude-verb

A category-governed grammar is *based* on categories, but there is more to being category-governed. The categories must also *govern* the modes of construction. The following is a proposed account of category-governed grammars.

In a *category-governed grammar*, no *significant* item figures in a grammatical construction *except in virtue of* its category.
No construction-rule pertains to an item *per se*, but only *per generis*.

7. Category-Based Reformulation of SL

We next observe that Example 1 is not category-governed, since significant items of SL – namely, the logical connectives – are not *officially* assigned categories.³

On the other hand, we can reformulate SL as a category-governed grammar as follows.

1. Categories

- (1) formulas (sentences)
- (2) 1-place connectives
- (3) 2-place connectives
- (4) nothing else

2. 1-Place Connectives

- (1) ‘~’ is a 1-place connective;
- (2) nothing else is a 1-place connective.

3. 2-Place Connectives

- (1) ‘&’ is a 2-place connective;
- (2) ‘∨’ is a 2-place connective;
- (3) ‘→’ is a 2-place connective;
- (4) ‘↔’ is a 2-place connective;
- (5) nothing else is a 2-place connective.

4. Formulas

- (1) every upper-case Roman letter is a formula;
- (2) a 1-place connective followed by a formula is a formula;
- (3) a left-parenthesis followed by a formula followed by a 2-place connective followed by a formula followed by a right-parenthesis is a formula;
- (4) nothing else is a formula.

³ The punctuation marks are not assigned categories either, but they are not "significant". The exact distinction between significant and insignificant is not trivial, but most people sense the difference.

8. Type-Governed (Categorial) Grammars

A special sort of category-governed grammars are *categorial grammars*. The usage is not entirely fortunate, since the name does not suggest how categorial grammars are specifically different from the general class of category-governed grammars. As an alternative name, we propose *type-governed* grammars.

What makes type-governed (i.e., categorial) grammars special is that the categories they propose are *recursively* constructed in a manner reminiscent of type-theory. In particular, a categorial grammar proposes an initial finite set of *primitive categories*, and it proposes finitely-many rules for constructing derivative categories, in virtue of which the grammar proposes infinitely-many *categories*.

We frequently use the term ‘type’ in reference to such categories, in order to distinguish them from traditional categories, such as NP and VP. We nevertheless continue to use the term ‘categorial’ as the associated adjective, especially to refer to type-based matters.

According to the scheme proposed by Kazimierz Adjukiewicz,⁴ categorial grammar is based on the following primitive types.

nouns
sentences

An immediate problem arises, since nouns ultimately divide into two sub-categories – namely:⁵

proper-nouns e.g., ‘Jay’, ‘Kay’
common-nouns e.g., ‘dog’, ‘cat’

Accordingly, many authors propose two primitive noun-categories. In this connection, we use the following notation.⁶

D definite-nouns⁷
C common-nouns
S sentences

In addition to primitive types, there are derivative types, which correspond to **functors**. Categorial grammar categorizes each functor according to:

- (1) what types of phrases it takes as **input**
- (2) what type of phrase it produces as **output**

The notation for functor-types varies from author to author. We propose the following.

(\mathfrak{I}_1	×	...	×	\mathfrak{I}_k)	→	\mathfrak{I}_0
takes	an expression of this type	and	...	and	an expression of this type	and	delivers	an expression of this type

⁴ Adjukiewicz, K., "Die Syntaktische Konnexität", *Studia Philosophica*. 1(1935), 1-27.

⁵ This is more obvious to a native speaker of English than it would be to a native speaker of Polish. In Polish, but not English, common nouns serve as subjects and objects of verbs, just like proper nouns.

⁶ This parallels David Lewis ["General Semantics", *Synthese*, 22 (1970), 18-67] who proposes types n, c, and s.

⁷ Often we use the term ‘name’ in reference to this category, which corresponds roughly to the term ‘singular-term’ in logic. Basically, this type applies to any expression that behaves – *syntactically and semantically* – like a *proper noun*. Note carefully that this is not a purely-syntactic category, since semantic criteria are partly used in identifying which phrases have this type.

The following is an alternative more common notation, which we will not employ.

(\mathfrak{J}_0	/	\mathfrak{J}_1	,	...	,	\mathfrak{J}_k)
delivers	an expression of this type	given	an expression of this type	and	...	and	an expression of this type		

9. Official (Inductive) Definition of Categories/Types

- (1) D is a type;
C is a type;
S is a type;
- (2) if A_1, \dots, A_k ($k \geq 1$) are types, then $((A_1 \times \dots \times A_k) \rightarrow A_0)$ is a type;
- (3) nothing else is a type.

Shorthand notation: $\mathcal{A}^2 =_{df} \mathcal{A} \times \mathcal{A}$
 $\mathcal{A}^3 =_{df} \mathcal{A} \times \mathcal{A} \times \mathcal{A}$
 etc.

10. Examples of Elementary Types

a k-place	has type	which is to say it is an expression, with k-many blanks, that...
Connective	$S^k \rightarrow S$	yields a sentence whenever these blanks are all filled by sentences .
Predicate	$D^k \rightarrow S$	yields a sentence whenever these blanks are filled by names .
Function-Sign	$D^k \rightarrow D$	yields a name whenever these blanks are filled by names .
Subnective	$S^k \rightarrow D$	yields a name whenever these blanks are filled by sentences

11. More Complex Examples

Example 1: $D \rightarrow (D \rightarrow S)$

A functor of this type takes a name (D) as input and generates an item of category $D \rightarrow S$ – i.e., a 1-place predicate – as output.

Example 2: $(D \rightarrow S) \rightarrow S$

This is an example of a *second-order* type. A functor of this type takes a 1-place predicate ($D \rightarrow S$) as input, and generates a sentence (S) as output.

Example 3: $(D \rightarrow S) \rightarrow (D \rightarrow S)$

This is also a second-order type. A functor of this type takes a 1-place predicate ($D \rightarrow S$) as input and generates a 1-place predicate ($D \rightarrow S$) as output.

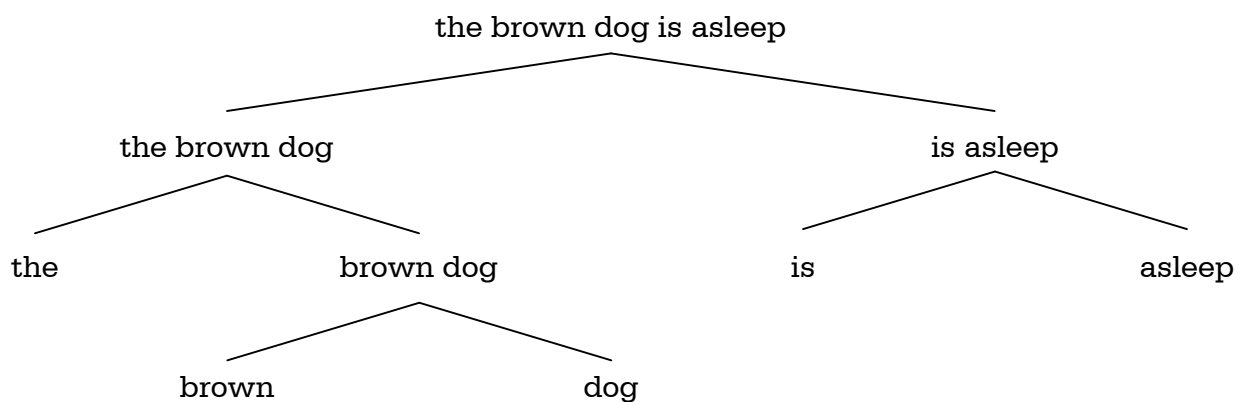
Example 4: $((D \rightarrow S) \rightarrow S) \rightarrow S$

This is an example of a *third-order* type. A functor of this type takes an expression of type $(D \rightarrow S) \rightarrow S$ as input and generates a sentence (S) as output.

12. Examples from English

1. Example 1: the brown dog is asleep

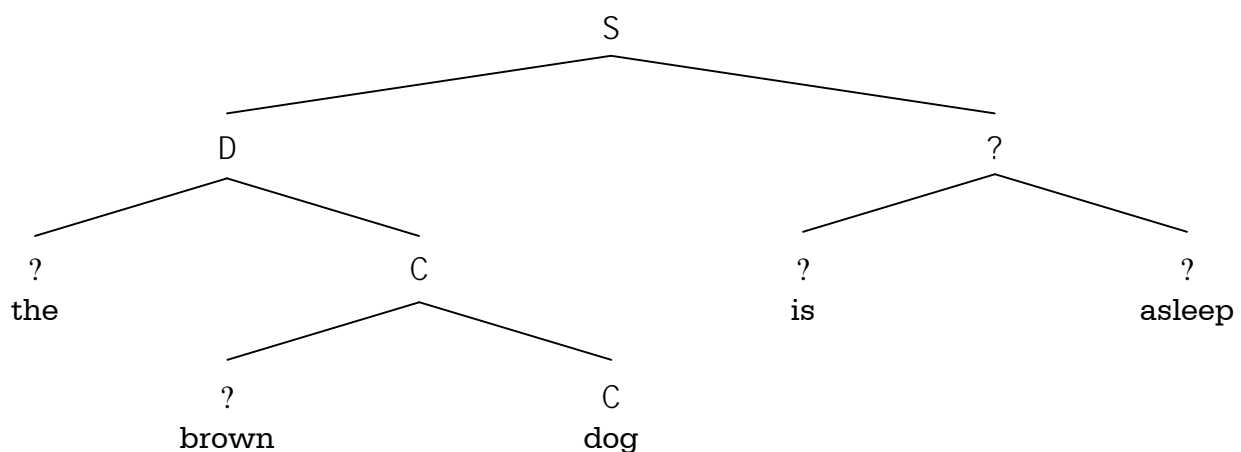
The following is a plausible parse-tree.



In proposing this parsing, we are in particular proposing that each node is a *constituent*, which is to say a (relatively) autonomous meaningful unit.

1. the top node is clearly a sentence, so its category is S.
2. ‘the brown dog’ behaves syntactically and semantically like a proper-noun [e.g., ‘Fido’], so it is plausibly assigned type D.
3. ‘dog’ is a paradigm common-noun, so we assign it type C.
4. ‘brown dog’ behaves very much like ‘dog’, so it plausibly assigned the same type – C.

So far, we have the following type analysis.



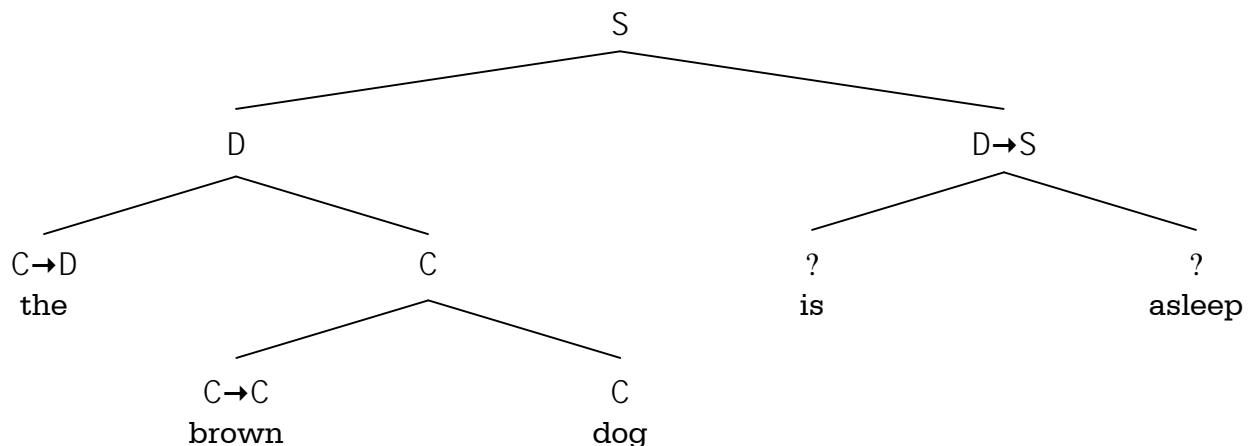
The remaining type assignments can now be calculated in accordance with the **basic categorial rule**, which is the following.

if \mathcal{E}_1 has type \mathfrak{I}_1 ,
and \mathcal{E}_2 has type $\mathfrak{I}_1 \rightarrow \mathfrak{I}_2$,
then \mathcal{E}_1 and \mathcal{E}_2 combine to form an expression of type \mathfrak{I}_2 .

5. ‘brown dog’ is composed out of ‘brown’ and ‘dog’. Since ‘dog’ is a C, which is a primitive type, ‘dog’ is not a functor. So ‘brown’ acts as a functor that applies to the C ‘dog’ to produce the C ‘brown dog’. Accordingly, ‘brown’ has type $C \rightarrow C$.
6. ‘the brown dog’ is composed out of ‘the’ and ‘brown dog’. Since ‘brown dog’ is a C, which is a primitive type, it is not a functor, so ‘the’ must be a functor. In particular, ‘the’ takes the C ‘brown dog’ and produces the D ‘the brown dog’. Accordingly, ‘the’ has type $C \rightarrow D$.

7. ‘the brown dog is asleep’ is composed out of ‘the brown dog’ and ‘is asleep’. Since the former has a primitive type, the latter must be a functor. In particular, ‘is asleep’ takes the D ‘the brown dog’ and produces the S ‘the brown dog is asleep’, and accordingly has type $D \rightarrow S$.

So we now have the following type analysis.

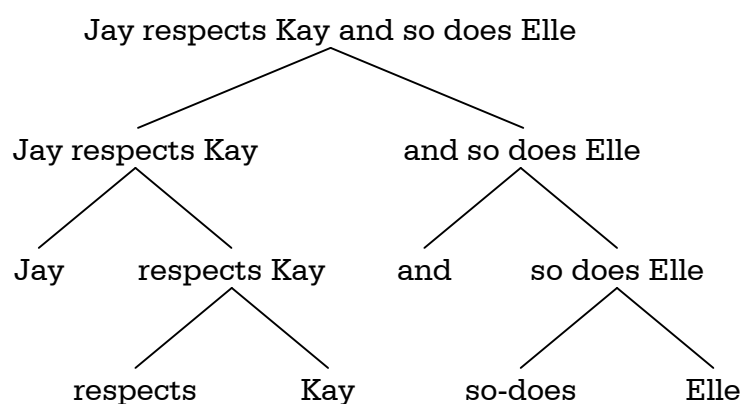


What remains is to provide a categorial analysis of ‘is’ and ‘asleep’ that enables us to construct ‘is asleep’ out of them. We postpone this analysis until later.

2. Example 2

Jay respects Kay, and so does Elle

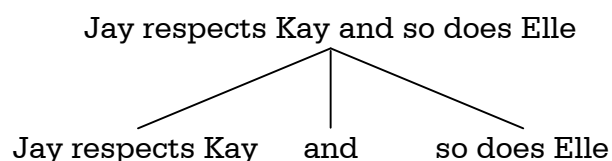
We propose the following parse,



Note that we treat ‘so does’ as a **morpheme** (smallest meaningful unit). English is replete with expressions that are phonetically-complex but semantically-simple.

Note also that we treat ‘respects Kay’ as a constituent. There are numerous tests that suggest that this phrase satisfies that suggest that it is understood as a unit. For example, in this very example, the phrase ‘so does’ is **anaphoric** to it. When we semantically process ‘so does’, we look for an earlier phrase that it *alludes* to; in this case, that phrase is ‘respects Kay’.

Finally, note that we treat ‘and so does Elle’ as a constituent. There is no overwhelming argument for this; the main argument is that it allows our tree to have a simplified (**binary**) structure. The following is an alternative analysis that is more in keeping with logic.

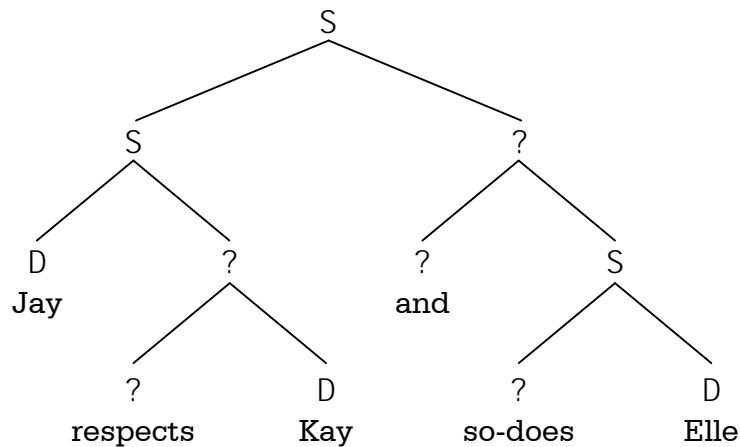


Analysis:

The following type-assignments are apparent.

- | | | |
|-----|-----------------------------------|---|
| (1) | Jay | D |
| (2) | Kay | D |
| (3) | Elle | D |
| (4) | Jay respects Kay | S |
| (5) | so does Elle | S |
| (6) | Jay respects Kay and so does Elle | S |

This yields the following provisional tree-analysis.



The rest is simply a matter of solving a category-logic puzzle.

- 'Jay', which has type D, combines with 'respects Kay', to form a sentence. Our categorial rules only permit one way of accomplishing this – 'respects Kay' must have type $D \rightarrow S$; it is a one-place predicate [what linguists call a 'verb phrase'].
- 'respects' combines with 'Kay' [type D] to form a one-place predicate [$D \rightarrow S$]; given our categorial rules, this means that 'respects' has type $D \rightarrow (D \rightarrow S)$.
- 'so does' combines with 'Elle' [type D] to form a sentence, so 'so does' is – like its antecedent – a verb phrase [type $D \rightarrow S$].
- 'and so does Elle' combines with 'Jay respects Kay' (a sentence) to form a larger sentence, so its type must be $S \rightarrow S$; it is a one-place connective.
- 'and' combines with 'so does Elle' [type S] to form item (4), which is a one-place connective [$S \rightarrow S$], so its type must be $S \rightarrow (S \rightarrow S)$.

Thus, we have the following final analysis.

