עיבוד שפות טבעיות

שולי וינטנר

Context-free grammars

A **context-free grammar** (CFG) is a four-tuple $\langle \Sigma, V, S, P \rangle$, where:

- Σ is a finite, non-empty set of **terminals**, the alphabet;
- V is a finite, non-empty set of **grammar variables** (categories, or non-terminal symbols), such that $\Sigma \cap V = \emptyset$;
- $S \in V$ is the **start symbol**;
- P is a finite set of **production rules**, each of the form $A \to \alpha$, where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$.

For a rule $A \to \alpha$, A is the rule's **head** and α is its **body**.

Context-free grammars: example

$\Sigma = \{\mathit{the}, \mathit{cat}, \mathit{in}, \mathit{hat}\}$

$V = \{D, N, P, NP, PP\}$

The start symbol is *NP*

The rules:

$$\begin{array}{ll} D \rightarrow \mathit{the} & NP \rightarrow D\ N \\ N \rightarrow \mathit{cat} & PP \rightarrow P\ NP \\ N \rightarrow \mathit{hat} & NP \rightarrow NP\ PP \\ P \rightarrow \mathit{in} & \end{array}$$

Context-free grammars: language

Each non-terminal symbol in a grammar denotes a language.

A rule such as $N \rightarrow cat$ implies that the language denoted by the non-terminal N includes the alphabet symbol *cat*.

The symbol *cat* here is a single, atomic alphabet symbol, and not a string of symbols: the alphabet of this example consists of natural language words, not of natural language letters.

For a more complex rule such as $NP \rightarrow D\ N$, the language denoted by NP contains the concatenation of the language denoted by D with that denoted by N: $L(NP) = L(D) \cdot L(N)$.

Matters become more complicate when we consider recursive rules such as $NP \rightarrow NP\ PP$.

Context-free grammars: derivation

Given a grammar $G = \langle V, \Sigma, P, S \rangle$, we define the set of *forms* to be $(V \cup \Sigma)^*$: the set of all sequences of terminal and non-terminal symbols.

Derivation is a relation that holds between two forms, each a sequence of grammar symbols.

A form α *derives* a form β, denoted by $\alpha \Rightarrow \beta$, if and only if $\alpha = \gamma_l A \gamma_r$ and $\beta = \gamma_l \gamma_c \gamma_r$ and $A \rightarrow \gamma_c$ is a rule in P.

A is called the *selected symbol*. The rule $A \rightarrow \gamma$ is said to be **applicable** to α.

Derivation: example

The set of non-terminals of G is $V = \{D, N, P, NP, PP\}$ and the set of terminals is $\Sigma = \{the, cat, in, hat\}$.

The set of forms therefore contains all the (infinitely many) sequences of elements from V and Σ, such as $\langle\rangle$, $\langle NP\rangle$, $\langle D\ cat\ P\ D\ hat\rangle$, $\langle D\ N\rangle$, $\langle the\ cat\ in\ the\ hat\rangle$, etc.

Let us start with a simple form, $\langle NP\rangle$. Observe that it can be written as $\gamma_l NP \gamma_r$, where both γ_l and γ_r are empty. Observe also that *NP* is the head of some grammar rule: the rule $NP \rightarrow D\ N$. Therefore, the form is a good candidate for derivation: if we replace the selected symbol *NP* with the body of the rule, while preserving its environment, we get $\gamma_l D\ N \gamma_r = D\ N$. Therefore, $\langle N\rangle \Rightarrow \langle D\ N\rangle$.

Derivation: example

We now apply the same process to $\langle D\ N \rangle$. This time the selected symbol is D (we could have selected N, of course). The left context is again empty, while the right context is $\gamma_r = N$. As there exists a grammar rule whose head is D, namely $D \rightarrow$ *the*, we can replace the rule's head by its body, preserving the context, and obtain the form $\langle \textit{the}\ N \rangle$. Hence $\langle D\ N \rangle \Rightarrow \langle \textit{the}\ N \rangle$.

Derivation: example

Given the form $\langle \mathit{the}\ N \rangle$, there is exactly one non-terminal that we can select, namely N. However, there are two rules that are headed by N: $N \rightarrow \mathit{cat}$ and $N \rightarrow \mathit{hat}$. We can select either of these rules to show that both $\langle \mathit{the}\ N \rangle \Rightarrow \langle \mathit{the\ cat} \rangle$ and $\langle \mathit{the}\ N \rangle \Rightarrow \langle \mathit{the\ hat} \rangle$.

Since the form $\langle \mathit{the\ cat} \rangle$ consists of terminal symbols only, no non-terminal can be selected and hence it derives no form.

Extended derivation

$\alpha \overset{k}{\Rightarrow}_G \beta$ if α derives β in k steps: $\alpha \Rightarrow_G \alpha_1 \Rightarrow_G \alpha_2 \Rightarrow_G \ldots \Rightarrow_G \alpha_k$ and $\alpha_k = \beta$.

The reflexive-transitive closure of '$\Rightarrow_G$' is '$\overset{*}{\Rightarrow}_G$': $\alpha \overset{*}{\Rightarrow}_G \beta$ if $\alpha \overset{k}{\Rightarrow}_G \beta$ for some $k \geq 0$.

A G-**derivation** is a sequence of forms $\alpha_1, \ldots, \alpha_n$, such that for every $i, 1 \leq i < n$, $\alpha_i \Rightarrow_G \alpha_{i+1}$.

Extended derivation: example

$$\begin{array}{llll}(1) & \langle NP \rangle & \Rightarrow & \langle D\ N \rangle \\ (2) & \langle D\ N \rangle & \Rightarrow & \langle the\ N \rangle \\ (3) & \langle the\ N \rangle & \Rightarrow & \langle the\ cat \rangle\end{array}$$

Therefore, we trivially have:

Extended derivation: example

$$
\begin{array}{llcl}
(4) & \langle NP \rangle & \stackrel{*}{\Rightarrow} & \langle D\ N \rangle \\
(5) & \langle D\ N \rangle & \stackrel{*}{\Rightarrow} & \langle the\ N \rangle \\
(6) & \langle the\ N \rangle & \stackrel{*}{\Rightarrow} & \langle the\ cat \rangle
\end{array}
$$

From (2) and (6) we get

$$(7) \quad \langle D\ N \rangle \stackrel{*}{\Rightarrow} \langle the\ cat \rangle$$

and from (1) and (7) we get

$$(7) \quad \langle NP \rangle \stackrel{*}{\Rightarrow} \langle the\ cat \rangle$$

Languages

A form α is a **sentential form** of a grammar G iff $S \overset{*}{\Rightarrow}_G \alpha$, i.e., it can be derived in G from the start symbol.

The (formal) **language** generated by a grammar G with respect to a category name (non-terminal) A is $L_A(G) = \{w \mid A \overset{*}{\Rightarrow} w\}$. The language generated by the grammar is $L(G) = L_S(G)$.

A language that can be generated by some CFG is a *context-free language* and the class of context-free languages is the set of languages every member of which can be generated by some CFG. If no CFG can generate a language L, L is said to be *trans-context-free*.

Language of a grammar

For the example grammar (with NP the start symbol):

$$\begin{array}{ll} D \rightarrow \mathit{the} & NP \rightarrow D\ N \\ N \rightarrow \mathit{cat} & PP \rightarrow P\ NP \\ N \rightarrow \mathit{hat} & NP \rightarrow NP\ PP \\ P \rightarrow \mathit{in} & \end{array}$$

it is fairly easy to see that $L(D) = \{\mathit{the}\}$.

Similarly, $L(P) = \{\mathit{in}\}$ and $L(N) = \{\mathit{cat},\ \mathit{hat}\}$.

Language of a grammar

It is more difficult to define the languages denoted by the non-terminals *NP* and *PP*, although is should be straight-forward that the latter is obtained by concatenating $\{\mathit{in}\}$ with the former.

Proposition: $L(\mathit{NP})$ is the denotation of the regular expression

$$\mathit{the} \cdot (\mathit{cat} + \mathit{hat}) \cdot (\mathit{in} \cdot \mathit{the} \cdot (\mathit{cat} + \mathit{hat}))^*$$

Language: a formal example G_e

$$\begin{array}{lcl} S & \rightarrow & V_a \;\; S \;\; V_b \\ S & \rightarrow & \epsilon \\ V_a & \rightarrow & a \\ V_b & \rightarrow & b \end{array}$$

$L(G_e) = \{a^n b^n \mid n \geq 0\}$.

Recursion

The language $L(G_e)$ is *infinite*: it includes an infinite number of words; G_e is a finite grammar.

To be able to produce infinitely many words with a finite number of rules, a grammar must be *recursive*: there must be at least one rule whose body contains a symbol, from which the head of the rule can be derived.

Put formally, a grammar $\langle \Sigma, V, S, P \rangle$ is recursive if there exists a chain of rules, $p_1, \ldots, p_n \in P$, such that for every $1 < i \leq n$, the head of p_{i+1} occurs in the body of p_i, and the head of p_1 occurs in the body of p_n.

In G_e, the recursion is simple: the chain of rules is of length 0, namely the rule $S \rightarrow V_a \;\; S \;\; V_b$ is in itself recursive.

Derivation tree

Sometimes derivations provide more information than is actually needed. In particular, sometimes two derivations of the same string differ not in the rules that were applied but only in the order in which they were applied.

Starting with the form $\langle \mathit{NP} \rangle$ it is possible to derive the string *the cat* in two ways:

$$(1) \quad \langle \mathit{NP} \rangle \Rightarrow \langle \mathit{D\ N} \rangle \Rightarrow \langle \mathit{D\ cat} \rangle \Rightarrow \langle \mathit{the\ cat} \rangle$$
$$(2) \quad \langle \mathit{NP} \rangle \Rightarrow \langle \mathit{D\ N} \rangle \Rightarrow \langle \mathit{the\ N} \rangle \Rightarrow \langle \mathit{the\ cat} \rangle$$

Since both derivations use the same rules to derive the same string, it is sometimes useful to collapse such "equivalent" derivations into one. To this end the notion of *derivation trees* is introduced.

Derivation tree

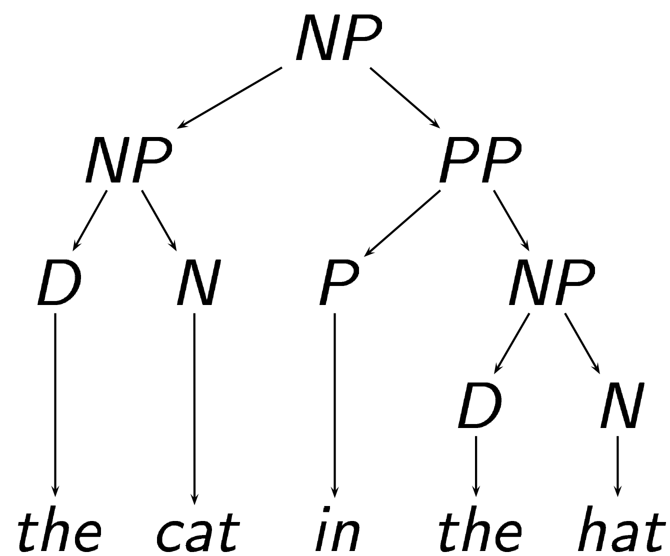
A derivation tree (sometimes called *parse* tree, or simply tree) is a visual aid in depicting derivations, and a means for imposing structure on a grammatical string.

Trees consist of vertices and branches; a designated vertex, the *root* of the tree, is depicted on the top. Then, branches are simply connections between two vertices.

Intuitively, trees are depicted “upside down”, since their root is at the top and their leaves are at the bottom.

Derivation tree: example

An example for a derivation tree for the string *the cat in the hat*:



Derivation tree

Formally, a tree consists of a finite set of vertices and a finite set of branches (or arcs), each of which is an ordered pair of vertices.

In addition, a tree has a designated vertex, the *root*, which has two properties: it is not the target of any arc, and every other vertex is accessible from it (by following one or more branches).

When talking about trees we sometimes use family notation: if a vertex v has a branch leaving it which leads to some vertex u, then we say that v is the *mother* of u and u is the *daughter*, or *child*, of v. If u has two daughters, we refer to them as *sisters*.

Derivation trees

Derivation trees are defined with respect to some grammar G, and must obey the following conditions:

1. every vertex has a *label*, which is either a terminal symbol, a non-terminal symbol or ϵ;

2. the label of the root is the start symbol;

3. if a vertex v has an outgoing branch, its label must be a non-terminal symbol, the head of some grammar rule; and the elements in body of the same rule must be the labels of the children of v, in the same order;

4. if a vertex is labeled ϵ, it is the only child of its mother.

Derivation trees

A *leaf* is a vertex with no outgoing branches.

A tree induces a natural "left-to-right" order on its leaves; when read from left to right, the sequence of leaves is called the *frontier*, or *yield* of the tree.

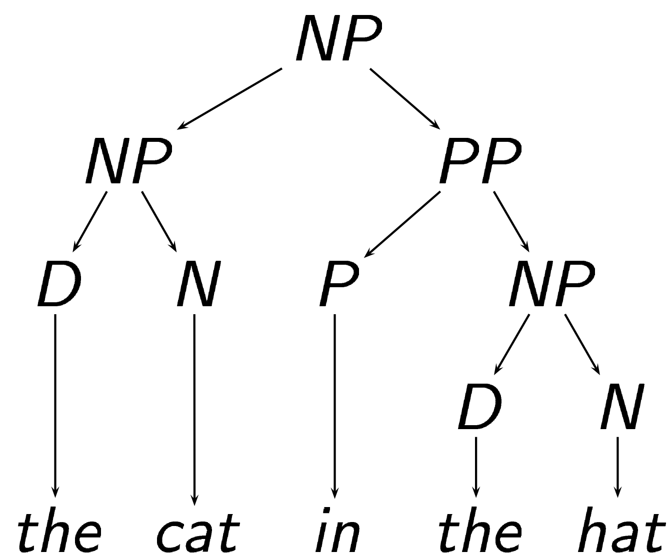
Correspondence between trees and derivations

Derivation trees correspond very closely to derivations.

For a form α, a non-terminal symbol A derives α if and only if α is the yield of some parse tree whose root is A.

Sometimes there exist different derivations of the same string that correspond to a single tree. In fact, the tree representation collapses exactly those derivations that differ from each other only in the order in which rules are applied.

Correspondence between trees and derivations



Each non-leaf vertex in the tree corresponds to some grammar rule (since it must be labeled by the head of some rule, and its children must be labeled by the body of the same rule).

Correspondence between trees and derivations

This tree represents the following derivations (among others):

(1) *NP* ⇒ *NP PP* ⇒ *D N PP* ⇒ *D N P NP*
⇒ *D N P D N* ⇒ *the N P D N*
⇒ *the cat P D N* ⇒ *the cat in D N*
⇒ *the cat in the N* ⇒ *the cat in the hat*

(2) *NP* ⇒ *NP PP* ⇒ *D N PP* ⇒ *the N PP*
⇒ *the cat PP* ⇒ *the cat P NP*
⇒ *the cat in NP* ⇒ *the cat in D N*
⇒ *the cat in the N* ⇒ *the cat in the hat*

(3) *NP* ⇒ *NP PP* ⇒ *NP P NP* ⇒ *NP P D N*
⇒ *NP P D hat* ⇒ *NP P the hat*
⇒ *NP in the hat* ⇒ *D N in the hat*
⇒ *D cat in the hat* ⇒ *the cat in the hat*

Correspondence between trees and derivations

While exactly the same rules are applied in each derivation (the rules are uniquely determined by the tree), they are applied in different orders. In particular, derivation (2) is a *leftmost* derivation: in every step the leftmost non-terminal symbol of a derivation is expanded. Similarly, derivation (3) is *rightmost*.

Ambiguity

Sometimes, however, different derivations (of the same string!) correspond to different trees.

This can happen only when the derivations differ in the rules which they apply.

When more than one tree exists for some string, we say that the string is *ambiguous*.

Ambiguity is a major problem when grammars are used for certain formal languages, in particular programming languages. But for natural languages, ambiguity is unavoidable as it corresponds to properties of the natural language itself.

Ambiguity: example

Consider again the example grammar and the following string:

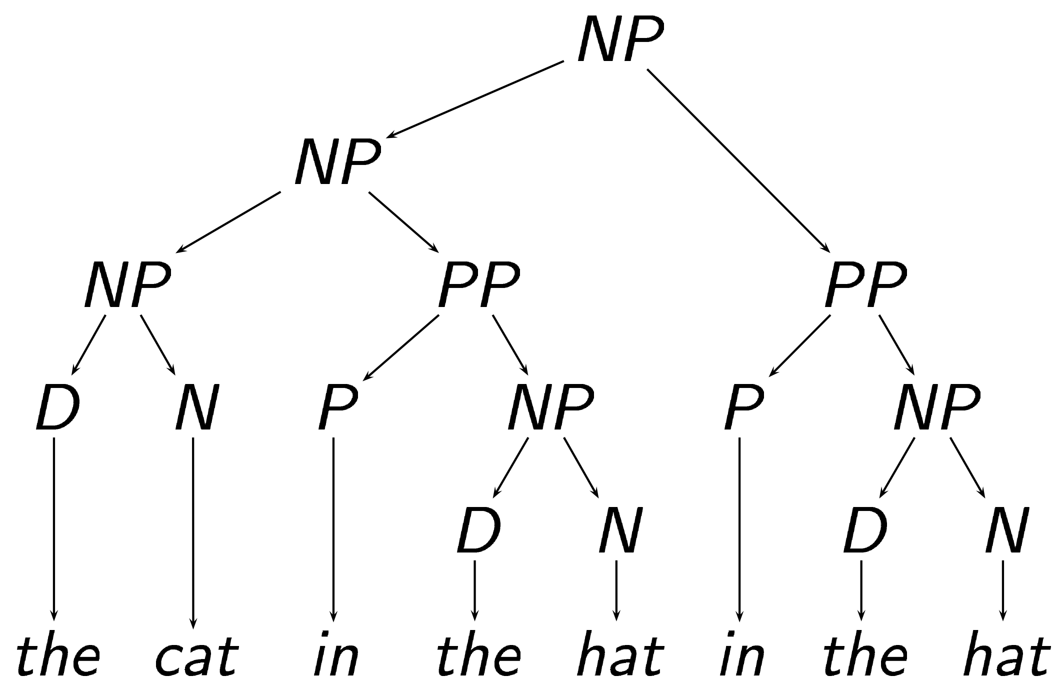
the cat in the hat in the hat

Intuitively, there can be (at least) two readings for this string: one in which a certain cat wears a hat-in-a-hat, and one in which a certain cat-in-a-hat is inside a hat:

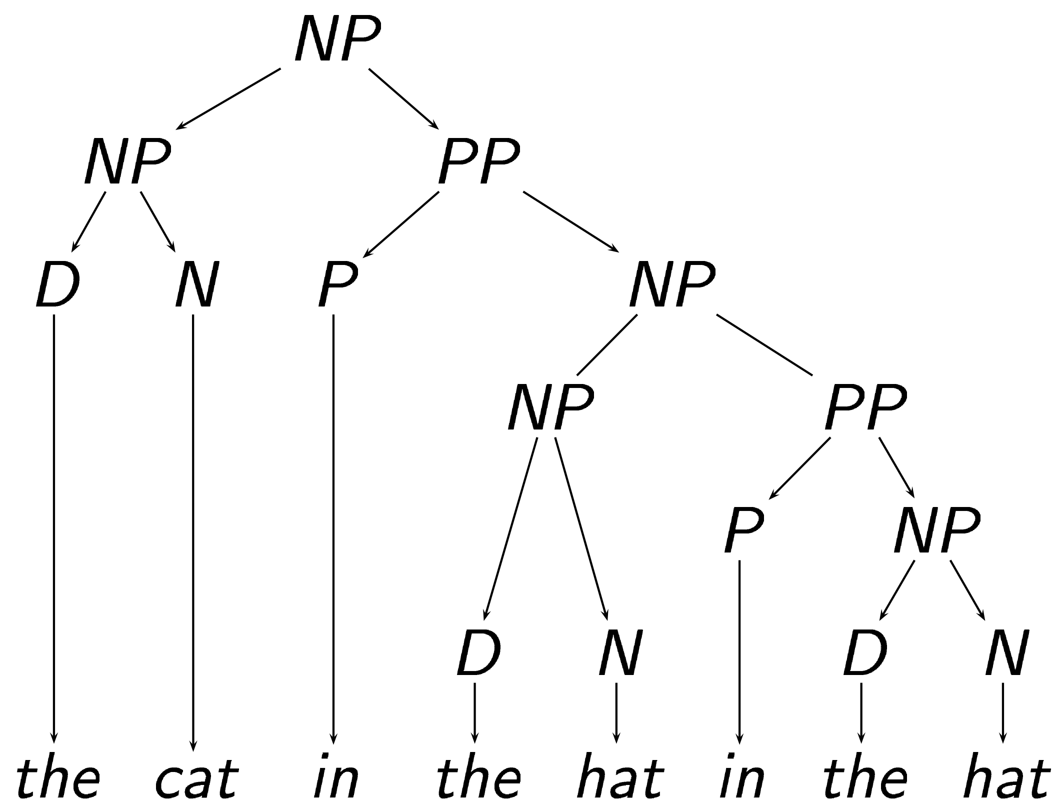
((the cat in the hat) in the hat)
(the cat in (the hat in the hat))

This distinction in intuitive meaning is reflected in the grammar, and hence two different derivation trees, corresponding to the two readings, are available for this string:

Ambiguity: example



Ambiguity: example



Ambiguity: example

Using linguistic terminology, in the left tree the second occurrence of the prepositional phrase *in the hat* modifies the noun phrase *the cat in the hat*, whereas in the right tree it only modifies the (first occurrence of) the noun phrase *the hat*. This situation is known as *syntactic* or *structural* ambiguity.

Context-free grammars for natural languages

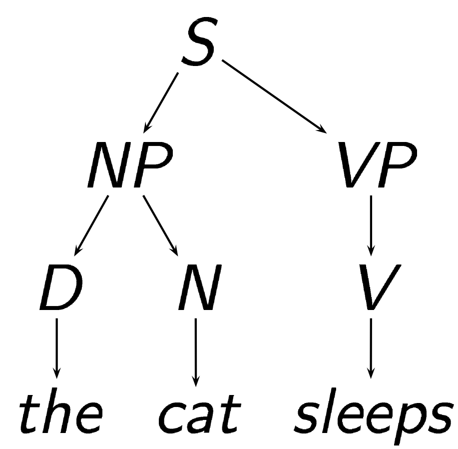
A context-free grammar for English sentences: $G = \langle V, \Sigma, P, S \rangle$ where $V = \{D, N, P, NP, PP, V, VP, S\}$, $\Sigma = \{the, cat, in, hat\}$, the start symbol is S and P is the following set of rules:

$$
\begin{array}{ll}
S \rightarrow NP\ VP & D \rightarrow the \\
NP \rightarrow D\ N & N \rightarrow cat \\
NP \rightarrow NP\ PP & N \rightarrow hat \\
PP \rightarrow P\ NP & V \rightarrow sleeps \\
VP \rightarrow V & P \rightarrow in \\
VP \rightarrow VP\ NP & V \rightarrow smile \\
VP \rightarrow VP\ PP & V \rightarrow loves \\
 & V \rightarrow saw
\end{array}
$$

Context-free grammars for natural languages

The augmented grammar can derive strings such as *the cat sleeps* or *the cat in the hat saw the hat*.

A derivation tree for *the cat sleeps* is:



Context-free grammars for natural languages

There are two major problems with this grammar.

1. it ignores the valence of verbs: there is no distinction among subcategories of verbs, and an intransitive verb such as *sleep* might occur with a noun phrase complement, while a transitive verb such as *love* might occur without one. In such a case we say that the grammar *overgenerates*: it generates strings that are not in the intended language.

2. there is no treatment of subject–verb agreement, so that a singular subject such as *the cat* might be followed by a plural form of verb such as *smile*. This is another case of overgeneration.

Both problems are easy to solve.

Verb valence

To account for valence, we can replace the non-terminal symbol *V* by a set of symbols: *Vtrans*, *Vintrans*, *Vditrans* etc. We must also change the grammar rules accordingly:

$VP \rightarrow Vintrans$	$Vintrnas \rightarrow sleeps$
$VP \rightarrow Vtrans\ NP$	$Vintrans \rightarrow smile$
$VP \rightarrow Vditrans\ NP\ PP$	$Vtrans \rightarrow loves$
	$Vditrans \rightarrow give$

Agreement

To account for agreement, we can again extend the set of non-terminal symbols such that categories that must agree reflect in the non-terminal that is assigned for them the features on which they agree. In the very simple case of English, it is sufficient to multiply the set of “nominal” and “verbal” categories, so that we get *Dsg*, *Dpl*, *Nsg*, *Npl*, *NPsg*, *NPpl*, *Vsg*, *Vlp*, *VPsg*, *VPpl* etc. We must also change the set of rules accordingly:

Agreement

$Nsg \rightarrow cat$	$Npl \rightarrow cats$
$Nsg \rightarrow hat$	$Npl \rightarrow hats$
$P \rightarrow in$	
$Vsg \rightarrow sleeps$	$Vpl \rightarrow sleep$
$Vsg \rightarrow smiles$	$Vpl \rightarrow smile$
$Vsg \rightarrow loves$	$Vpl \rightarrow love$
$Vsg \rightarrow saw$	$Vpl \rightarrow saw$
$Dsg \rightarrow a$	$Dpl \rightarrow many$

Agreement

$S \rightarrow NPsg\ VPsg$	$S \rightarrow NPpl\ VPpl$
$NPsg \rightarrow Dsg\ Nsg$	$NPpl \rightarrow Dpl\ Npl$
$NPsg \rightarrow NPsg\ PP$	$NPpl \rightarrow NPpl\ PP$
$PP \rightarrow P\ NP$	
$VPsg \rightarrow Vsg$	$VPpl \rightarrow Vpl$
$VPsg \rightarrow VPsg\ NP$	$VPpl \rightarrow VPpl\ NP$
$VPsg \rightarrow VPsg\ PP$	$VPpl \rightarrow VPpl\ PP$

Context-free grammars for natural languages

Context-free grammars can be used for a variety of syntactic constructions, including some non-trivial phenomena such as unbounded dependencies, extraction, extraposition etc.

However, some (formal) languages are not context-free, and therefore there are certain sets of strings that cannot be generated by context-free grammars.

The interesting question, of course, involves natural languages: are there natural languages that are not context-free? Are context-free grammars sufficient for generating every natural language?