angu l'gezel

ש Teens אתרי
For a rule $A \rightarrow \alpha$, $A$ is the rule's head and $\alpha$ is its body.

\[
\begin{align*}
&\mathbf{p} \quad \text{is a finite set of production rules, each of the form} \\
&\lambda \in \mathbf{S} \quad \text{is the start symbol,} \\
&\emptyset = \lambda \cup \Sigma \quad \text{such that} \\
&\Sigma \quad \text{is a finite, non-empty set of terminals, the alphabet,} \\
&\mathbf{I} \quad \text{is a finite, non-empty set of variables} \\
&\mathbf{X} \quad \text{is a finite, non-empty set of categories, or non-terminal symbols},
\end{align*}
\]

where:

\[
\langle \mathbf{I}, \mathbf{S}, \lambda, \mathbf{p} \rangle \quad \text{is a four-tuple (CFG) is a context-free grammar (CFG) is a context-free grammar}.
\]
The rules:

The start symbol is $NP$

\[
\{D, N, p, NP, pp\} = \{\text{the, cat, in, hat}\}
\]

Context-free grammars example
rules such as \( Np \rightarrow Np p \).

Matters become more complicated when we consider recursive
\((N)T \cdot (D)T = (Dp)T \) with that denoted by \( N \).

\( N \) contains the concatenation of the language
denoted by \( Np \) by the language
for a more complex rule such as \( N D \rightarrow Np \).

For natural language words, not of natural language letters,
not a string of symbols: the alphabet of this example consists

The symbol here is a single, atomic alphabet symbol, and

the non-terminal \( N \) includes the alphabet symbol \( cat \).

A rule such as \( N \rightarrow \) cat implies that the language denoted by

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

Context-free grammars: language
A is called the selected symbol. The rule $A \rightarrow \gamma$ is said to be applicable to $\alpha$.

A form $\alpha$ derives a form $\beta$, denoted by $\alpha \Rightarrow \beta$, if and only if $\beta$ is a sequence of grammar symbols.

A form $\alpha$ is a relation that holds between two forms, each a non-terminal symbol.

Given a grammar $G = \langle S, \Sigma, \Gamma, \delta \rangle$, we define the set of forms $\langle S, \Sigma, \Gamma \rangle^*$ to be (the set of all sequences of terminal and non-terminal symbols) $\{ \gamma \in \Sigma^* \mid S \Rightarrow^* \gamma \}$.

Context-free grammars: derivation.
\[ \langle N D \rangle \iff \langle N \rangle \text{ Therefore, } N \text{ is a } p \text{ NP } \text{ with the } \text{ body of the rule, while preserving its environment, we get} \]

\[ N D \iff N \text{ Therefore, the form is a good candidate for} \]

\[ \text{ derivation: if we replace the selected symbol } NP \text{ with the} \]

\[ Np \text{ is the head of some grammar rule: the rule} \]

\[ \text{ also that } Np \text{ are empty. Observe written as } Np \text{ where both } \text{ and } \text{ are empty. Observe} \]

\[ \text{ let us start with a simple form, } Np. \text{ Observe that it can be} \]

\[ \text{ the cat in the hat}, \text{ etc.} \]

\[ \langle d \text{ cat } p \text{ hat} \rangle, \langle d \rangle, \langle N \rangle \text{ the cat in the hat} \text{, etc.} \]

\[ \text{ sequences of elements from } \text{ such as } \text{ and } \text{ all the (infinitely many)} \]

\[ \text{ the set of terminals is } \text{ the set of non-terminals of } \text{ and } \text{ and} \]

\[ \text{ Derivation: example} \]
\[ \langle \text{the N} \rangle \leftrightarrow \langle \text{N D} \rangle \]

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

**Derivation: Example**
Since the form \( \text{the cat} \) consists of terminal symbols only, no non-terminal can be selected and hence it derives no form.

\[
\langle \text{the hat} \rangle \leftarrow \langle N \rangle \ \text{the cat} \leftarrow \langle N \rangle \ \text{cat} \leftarrow N \ \text{hat}
\]

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

**Derivation: Example**
for every $i$, $1 \leq i \leq n$.

A $\mathcal{C}$-derivation is a sequence of forms $\alpha_1, \ldots, \alpha_n$ such that

$$
\mathcal{C} \subseteq \mathcal{C} \subseteq \mathcal{C} \subseteq \cdots \subseteq \mathcal{C} \subseteq \mathcal{C}
$$

$\mathcal{C} \subseteq \mathcal{C}$ for some $k \geq 0$.

The reflexive-transitive closure of $\mathcal{C}$, $\mathcal{C}^*$, is

$$
\mathcal{C} \subseteq \mathcal{C} \subseteq \mathcal{C} \subseteq \cdots \subseteq \mathcal{C} \subseteq \mathcal{C}
$$

if $\mathcal{C} \subseteq \mathcal{C}$ and $\alpha \in \mathcal{C} \Rightarrow \alpha \in \mathcal{C}$.

Extended derivation
Therefore, we trivially have:

\[
\begin{align*}
\langle \text{the cat} \rangle & \iff \langle N \text{ the} \rangle \quad (3) \\
\langle \text{the} N \rangle & \iff \langle N \ P \rangle \quad (2) \\
\langle N \ P \rangle & \iff \langle N P \rangle \quad (1)
\end{align*}
\]

Extended derivation: example
\[ \langle \text{the cat} \rangle \leftrightarrow \langle pN \rangle \ (7) \]

and from (1) and (7) we get

\[ \langle \text{the cat} \rangle \leftrightarrow \langle N \ D \rangle \ (7) \]

From (2) and (6) we get

\[ \langle \text{the cat} \rangle \leftrightarrow \langle N \ \text{the} \rangle \ (6) \]
\[ \langle \text{the} \rangle \leftrightarrow \langle N \ D \rangle \ (5) \]
\[ \langle N \ D \rangle \leftrightarrow \langle pN \rangle \ (4) \]

Extended derivation: example
be trans-context-free.

Thus, a language \( L \) is said to be context-free if no CFG can generate it. Hence, the class of context-free languages is the free language and the class of context-free languages that can be generated by some CFG is a context-free language.

\[
(\mathcal{L})^S = (\mathcal{L})^V = \{ m \in A^* \mid \exists \{ \text{non-terminal}\} \text{ grammar with } \mathcal{L} \text{ generated} \}
\]

The language generated by a grammar \( \mathcal{G} \) with respect to a context-free grammar \( \mathcal{G} \) is \( \left( \mathcal{G} \right)^V \). The language can be derived from \( \mathcal{G} \) from the start symbol 

\( A \) is a sentential form of a grammar \( \mathcal{G} \) iff \( S \in A \).
\{ \text{cat, hat} \} = (\text{N}) \quad \text{and} \quad \{ \text{in} \} = (\text{D})

\{ \text{the} \} = (\text{D})

\text{it is fairly easy to see that} \quad \text{I(} p \text{)}

\text{in} \leftrightarrow \text{p}
\text{hat} \leftrightarrow \text{N}
\text{cat} \leftrightarrow \text{N}
\text{the} \leftrightarrow \text{D}

\text{For the example grammar (with \text{NP} the start symbol):}

\text{Language of a grammar}
The denotation of the regular expression 

\( (\text{cat} \cdot \hat{\text{cat}}) \cdot (\text{in} \cdot \text{the} \cdot (\text{cat} + \hat{\text{cat}})) \)

is the proposition: \( I(NP) \) is the denotation of the former.

Forwarding that the latter is obtained by concatenating \{in\} with \{np\} and \{pp\}, although it should be straightforward.

It is more difficult to define the languages denoted by the language of a grammar.
\[
\{ 0 \geq u \mid u \in \mathcal{L} \} = (\mathcal{L})^+
\]

Language: a formal example $\mathcal{L}$

$q \leftarrow qL$
$p \leftarrow pL$
$\varepsilon \leftarrow S$
$qL \ S \ pL \leftarrow S$

Natural Language Processing
namely the rule $S \Lambda \rightarrow S \Lambda$ is in itself recursive.

In $G^e$, the recursion is simple: the chain of rules is of length 0,

$u \rightarrow \cdots \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \right
trees is introduced. To this end the notion of derivation is introduced into one. To collapse such "equivalent" strings, it is sometimes useful to derive the same sentences. Both derivations use the same rules to derive the same sentence in two ways:

Starting with the form \( \text{NP} \), it is possible to derive the string \( \text{the cat} \).

The cat in two ways:

1. \( \text{NP} \leftarrow \text{NP} \leftarrow \text{NP} \leftarrow \text{NP} \)
2. \( \text{NP} \leftarrow \text{NP} \leftarrow \text{NP} \leftarrow \text{NP} \)

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

**Derivation tree**
is at the top and their leaves are at the bottom.

Intuitively, trees are depicted "upside down," since their root

are simply connections between two vertices. The root of the tree, is depicted on the top. Then, branches

and branches; a designated vertex,

implies structure on a grammatical string. A derivation tree (sometimes called parse tree, or simply

Derivation tree
Derivation tree: example

An example for a derivation tree for the string the cat in the hat.
them as sisters.

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

branches).

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

vertices.

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

Derivation tree
4. If a vertex is labeled $e$, it is the only child of its mother.

3. If a vertex $v$ has an outgoing branch, its label must be a non-terminal symbol or $e$.

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

1. Every vertex has a label, which is either a terminal symbol or a non-terminal symbol.

$G$, and must obey the following conditions:

Derivation trees are defined with respect to some grammar.
the frontier, or yield of the tree.

A tree induces a natural "left-to-right" order on its leaves.

A leaf is a vertex with no outgoing branches.

Derivation trees
From each other only in the order in which rules are applied.

Representation collapses exactly those derivations that differ
string that correspond to a single tree. In fact, the tree
sometimes there exist different derivations of the same

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

Derivation trees correspond very closely to derivations.

Correspondence between trees and derivations.
its children must be labeled by the body of the same rule)

Since it must be labeled by the head of some rule, and

Each non-leaf vertex in the tree corresponds to some grammar

Correspondence between trees and derivations
This tree represents the following derivations (among others):

Correspondence between trees and derivations:
Correspondence between trees and derivations

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

unavoidable as it corresponds to properties of the natural
languages. But for natural languages, ambiguity is
for certain formal languages, in particular programming
Ambiguity is a major problem when grammars are used
the string is ambiguous.
When more than one tree exists for some string, we say that
which they apply.
This can happen only when the derivations differ in the rules
string(s) correspond to different trees.
Sometimes, however, different derivations (or the same
Ambiguity
corresponding to the two readings, are available for this string:

the grammar, and hence two different derivation trees,

This distinction in intuitive meaning is reflected in

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

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

which a certain cat-in-a-hat is inside a hat:

one in which a certain cat wears a hat-in-a-hat, and one in

Intuitively, there can be (at least) two readings for this string:

the cat in the hat in the hat

Consider again the example grammar and the following string:

Ambiguity: example
Ambiguity: example
Ambiguity: example
Ambiguity.

This situation is known as syntactic or structural.

It only modifies the first occurrence of the noun phrase
the noun phrase the cat in the hat, whereas in the right tree
occurrence of the prepositional phrase in the hat modifies

Using linguistic terminology, in the left tree the second
Ambiguity: example
saw $\leftarrow \Lambda$
loves $\leftarrow \Lambda$
smile $\leftarrow \Lambda$
in $\leftarrow \mathbf{p}$
sleeps $\leftarrow \Lambda$
hat $\leftarrow \mathbf{N}$
cat $\leftarrow \mathbf{N}$
the $\leftarrow \mathbf{D}$

\[ \mathbf{Np} \leftarrow \mathbf{pp} \]
\[ \mathbf{pp} \leftarrow \mathbf{np} \]
\[ \mathbf{np} \leftarrow \mathbf{D} \]
\[ \mathbf{D} \leftarrow \Lambda \]

Following set of rules:

A context-free grammar for English sentences: $\mathbf{C} = \mathbf{S}$
A derivation tree for the cat sleeps is:

```
s -> vp
  |     
v -> N
  |   vp
N -> the cat
  |   sleeps
D ->pv
    |  
vp
```

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

Context-free grammars for natural language
Both problems are easy to solve.

overgeneralization.

plural form of verb such as *smile*. This is another case of

a singular subject such as the cat might be followed by a

2. there is no treatment of subject–verb agreement, so that

generates strings that are not in the intended language.

such a case we say that the grammar overgeneralizes: it

a transitive verb such as *love* might occur without one. In

as *sleep* might occur with a noun phrase complement, while

among subcategorizes of verbs, and an intransitive verb such

I. it ignores the valence of verbs: there is no distinction

There are two major problems with this grammar.

Context-Free Grammars for Natural Languages
Verb valence

\[
\text{Verb} \leftarrow \text{Vittrons} \\
\text{Vittrons} \leftarrow \text{loves} \\
\text{Vittrons} \leftarrow \text{smile} \\
\text{Vittrons} \leftarrow \text{sleeps} \\
\text{Vittrons} \leftarrow \text{Vittrons} \ 	ext{Vittrons} \\
\text{Vittrons} \leftarrow \text{Vittrons} \ 	ext{Vittrons} \\
\text{Vittrons} \leftarrow \text{Vittrons} \ 	ext{Vittrons} \\
\]

etc.

We must also change the grammar rules accordingly:

symbol \( \text{Vittrons} \) by a set of symbols: \( \text{Vittrons}, \text{Vittrons}, \text{Vittrons}, \text{Vittrons} \). To account for valence, we can replace the non-terminal

Verb valence
rules accordingly: Vsg, Vp, Vpsg, Vpsl etc. We must also change the set of categories, so that we get Dsg, Dpl, Nsg, Npl, Npsg, Npsl, is sufficient to multiply the set of „nominal“ and „verbal“ on which they agree. In the very simple case of English, it in the non-terminal that is assigned for them the features terminal symbols such that categories that must agree reflect Agreement.

To account for agreement, we can again extend the set of non-

Natural Language Processing

35
Agreement
\[dd \mid d \in \Lambda \] 
\[pp \mid p \in \Lambda \] 
\[Np \mid p \in \Lambda \] 
\[Id \mid d \in \Lambda \] 
\[S \mid d \in \Lambda \] 

Agreement
Every natural language is not context-free. Are context-free grammars sufficient for generating languages that are not context-free? Languages are the natural languages that involve natural languages generated by context-free grammars. Therefore, there are certain sets of strings that cannot be however, some (formal) languages are not context-free, and unbounded dependencies, extractions, extraposition, etc. Context-free grammars can be used for a variety of syntactic structures, including some non-trivial phenomena such as

Context-free grammars for natural languages