Formal Grammar

Languages & Grammars

Phrase-Structure Grammars

Types of Phrase-Structure Grammars

Derivation Trees

Backus-Naur Form

Intro to Languages

English grammar tells us if a given combination of words is a valid sentence.

The syntax of a sentence concerns its form while the semantics concerns its meaning.

e.g. the mouse wrote a poem

From a syntax point of view this is a valid sentence.

From a semantics point of view not so fast…perhaps in Disney land

Natural languages (English, French, Portguese, etc) have very complex rules of syntax and not necessarily well-defined.

Formal Language

Formal language – is specified by well-defined set of rules of syntax

We describe the sentences of a formal language using a grammar.

Two key questions:

1 - Is a combination of words a valid sentence in a formal language?

2 – How can we generate the valid sentences of a formal language?

Formal languages provide models for both natural languages and programming languages.

Grammars

- A formal *grammar G* is any compact, precise mathematical definition of a language *L*.
  - As opposed to just a raw listing of all of the language's legal sentences, or just examples of them.
- A grammar implies an algorithm that would generate all legal sentences of the language.
  - Often, it takes the form of a set of recursive definitions.
- A popular way to specify a grammar recursively is to specify it as a *phrase-structure grammar*.

Grammars (Semi-formal)

Example: A grammar that generates a subset of the English language

$$\langle sentence \rangle \rightarrow \langle noun\_phrase \rangle \ \langle predicate \rangle$$

$$\langle noun\_phrase \rangle \rightarrow \langle article \rangle \ \langle noun \rangle$$

$$\langle predicate \rangle \rightarrow \langle verb \rangle$$

$$\langle article \rangle \rightarrow a$$

$$\langle article \rangle \rightarrow the$$

$$\langle noun \rangle \rightarrow boy$$

$$\langle noun \rangle \rightarrow dog$$

$$\langle verb \rangle \rightarrow runs$$

$$\langle verb \rangle \rightarrow sleeps$$

A derivation of "the boy sleeps":

$$
\begin{aligned}
\langle sentence \rangle &\Rightarrow \langle noun\_phrase \rangle \ \langle predicate \rangle \\
&\Rightarrow \langle noun\_phrase \rangle \ \langle verb \rangle \\
&\Rightarrow \langle article \rangle \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow the \ \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow the \ \ boy \ \ \langle verb \rangle \\
&\Rightarrow the \ \ boy \ \ sleeps
\end{aligned}
$$

A derivation of "a dog runs":

$$
\begin{aligned}
\langle sentence \rangle &\Rightarrow \langle noun\_phrase \rangle \ \langle predicate \rangle \\
&\Rightarrow \langle noun\_phrase \rangle \ \langle verb \rangle \\
&\Rightarrow \langle article \rangle \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow a \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow a \ dog \ \langle verb \rangle \\
&\Rightarrow a \ dog \ runs
\end{aligned}
$$

Language of the grammar:

L = { "a boy runs",
"a boy sleeps",
"the boy runs",
"the boy sleeps",
"a dog runs",
"a dog sleeps",
"the dog runs",
"the dog sleeps" }

Notation

$$\langle noun \rangle \rightarrow boy$$

$$\langle noun \rangle \rightarrow dog$$

Variable
or
Non-terminal

Symbols of
the vocabulary

Production
rule

Terminal
Symbols of
the vocabulary

Basic Terminology

- A ***vocabulary***/***alphabet***, V is a finite nonempty set of elements called symbols.
  - Example: V = {a, b, c, A, B, C, S}
- A ***word***/***sentence*** over V is a string of finite length of elements of V.
  - Example: *Aba*
- The ***empty***/***null string***, λ is the string with no symbols.
- V^* is the set of all words over V.
  - Example: V^* = {*Aba, BBa, bAA, cab* ...}
- A ***language*** over V is a subset of V^*.
  - We can give some criteria for a word to be in a language.

Phrase-Structure Grammars

A *phrase-structure grammar* (abbr. PSG) $G = (V,T,S,P)$ is a 4-tuple, in which:

- V is a vocabulary (set of symbols)
  - The "template vocabulary" of the language.
- $T \subseteq V$ is a set of symbols called *terminals*
  - Actual symbols of the language.
  - Also, $N :\equiv V - T$ is a set of special "symbols" called *nonterminals*. (Representing concepts like "noun")
- $S \in N$ is a special nonterminal, the *start symbol*.
  - in our example the start symbol was "sentence".
- P is a set of *productions* (to be defined).
  - Rules for substituting one sentence fragment for another
  - Every production rule must contain at least one nonterminal on its left side.

Phrase-structure Grammar

EXAMPLE:

- Let $G = (V, T, S, P)$,

- where $V = \{a, b, A, B, S\}$
- $T = \{a, b\}$,
- S is a start symbol
- $P = \{S \to ABa, A \to BB, B \to ab, A \to Bb\}$.

G is a Phrase-Structure Grammar.

What sentences can be generated with this grammar?

Derivation

Definition

Let $G=(V,T,S,P)$ be a phrase-structure grammar.

Let $w_0=lz_0r$ (the concatenation of l, z_0, and r) $w_1=lz_1r$ be strings over V.

If $z_0 \rightarrow z_1$ is a production of G we say that w1 is directly derivable from w0 and we write $w_o \Rightarrow w_1$.

If $w_0, w_1, \ldots, w_n$ are strings over V such that $w_0 \Rightarrow w_1, w_1 \Rightarrow w_2, \ldots, w_{n-1} \Rightarrow w_n$, then we say that w_n is derivable from w_0, and write $w_0 \Rightarrow^* w_n$.

The sequence of steps used to obtain w_n from w_o is called a derivation.

Language

Let G(V,T,S,P) be a phrase-structure grammar. The language generated by G (or the language of G) denoted by L(G) , is the set of all strings of terminals that are derivable from the starting state S.

$$L(G)= \{w \in T^* \mid S \Rightarrow^* w\}$$

Language L(G)

- **EXAMPLE:**

Let $G = (V, T, S, P)$, where $V = \{a, b, A, S\}$, $T = \{a, b\}$, S is a start symbol and $P = \{S \to aA, S \to b, A \to aa\}$.

The language of this grammar is given by L (G) = $\{b, aaa\}$;

1. we can derive aA from using $S \to aA$, and then derive aaa using $A \to aa$.

2. We can also derive b using $S \to b$.

Another example

Grammar: G=(V,T,S,P) T={a,b} P = $S \to aSb$

$S \to \lambda$

V={a,b,S}

Derivation of sentence : ab

$$S \Rightarrow aSb \Rightarrow ab$$

$S \to aSb$ $S \to \lambda$

Grammar:

$$S \to aSb$$

$$S \to \lambda$$

Derivation of sentence $aabb$:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

$$S \to aSb \qquad\qquad S \to \lambda$$

Other derivations:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb$$

$$\Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$$

So, what's the language of the grammar with the productions?

$$S \to aSb$$

$$S \to \lambda$$

Language of the grammar with the productions:

$$S \to aSb$$

$$S \to \lambda$$

$$L = \{a^n b^n : n \geq 0\}$$

PSG Example – English Fragment

We have $G = (V, T, S, P)$, where:

V = {**(sentence)**, **(noun phrase)**,
(verb phrase), **(article)**, **(adjective)**,
(noun), **(verb)**, **(adverb)**, *a*, *the*, *large*,
hungry, *rabbit*, *mathematician*, *eats*, *hops*,
quickly, *wildly*}

T = {*a*, *the*, *large*, *hungry*, *rabbit*, *mathematician*,
eats, *hops*, *quickly*, *wildly*}

S = **(sentence)**

P = (see next slide)

Productions for our Language

P = { **(sentence)** → **(noun phrase) (verb phrase)**,
(noun phrase) → **(article) (adjective) (noun)**,
(noun phrase) → **(article) (noun)**,
(verb phrase) → **(verb) (adverb)**,
(verb phrase) → **(verb)**,
(article) → *a*, **(article)** → *the*,
(adjective) → *large*, **(adjective)** → *hungry*,
(noun) → *rabbit*, **(noun)** → *mathematician*,
(verb) → *eats*, **(verb)** → *hops*,
(adverb) → *quickly*, **(adverb)** → *wildly* }

A Sample Sentence Derivation

(sentence)

(noun phrase) (verb phrase)

(article) (adj.) (noun) (verb phrase)

(art.) (adj.) (noun) (verb) (adverb)

the **(adj.) (noun) (verb) (adverb)**

the large **(noun) (verb) (adverb)**

the large rabbit **(verb) (adverb)**

the large rabbit hops **(adverb)**

the large rabbit hops quickly

On each step, we apply a production to a fragment of the previous sentence template to get a new sentence template. Finally, we end up with a sequence of terminals (real words), **that is, a sentence of our language *L*.**

Another Example

Let $G = (\overbrace{\{a, b, A, B, S\}}^{V}, \overbrace{\{a, b\}}^{T}, S,$

$\overbrace{\{S \to ABa, A \to BB, B \to ab, AB \to b\}}^{P})$.

One possible derivation in this grammar is:

$$S \Rightarrow ABa \Rightarrow Aaba \Rightarrow BBaba \Rightarrow Bababa$$
$$\Rightarrow abababa.$$

Defining the PSG Types

Type 0: Phase-structure grammars – no restrictions on the production rules

Type 1: Context-Sensitive PSG:

- All after fragments are either longer than the corresponding before fragments, or empty:

  if b → a, then $|b| < |a| \quad \vee \quad a = \lambda$.

Type 2: Context-Free PSG:

- All before fragments have length 1 and are nonterminals:

  if b → a, then $|b| = 1$ ($b \in N$).

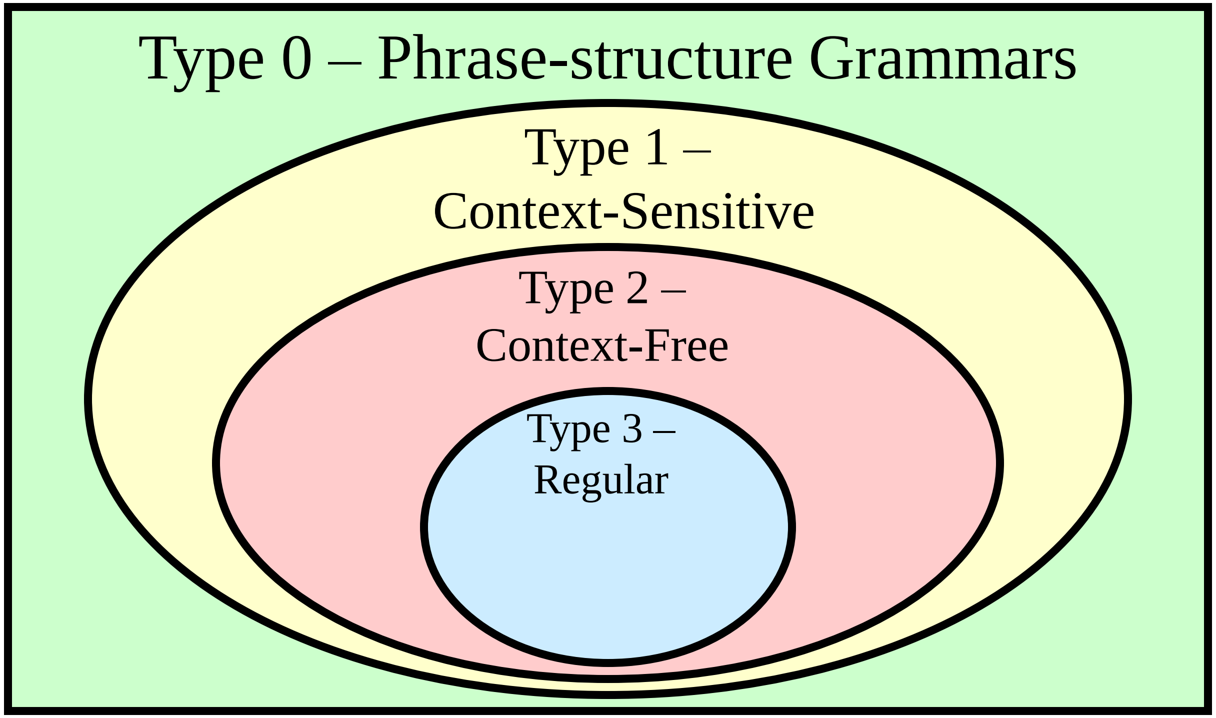
Type 3: Regular PSGs:

- All before fragments have length 1 and nonterminals
- All after fragments are either single terminals, or a pair of a terminal followed by a nonterminal.

  if b → a, then $a \in T \quad \vee \quad a \in TN$.

Types of Grammars - Chomsky hierarchy of languages

Venn Diagram of Grammar Types:



Classifying grammars

Given a grammar, we need to be able to find the smallest class in which it belongs. This can be determined by answering three questions:

Are the left hand sides of all of the productions single non-terminals?

If yes, does each of the productions create at most one non-terminal and is it on the right?

Yes – regular **No – context-free**

If not, can any of the rules reduce the length of a string of terminals and non-terminals?

Yes – unrestricted **No – context-sensitive**

Definition: Context-Free Grammars

Grammar $G = (V, T, S, P)$

Vocabulary

Terminal symbols

Start variable

Productions of the form:

$$A \rightarrow x$$

Non-Terminal

String of variables and terminals

Derivation Tree of A Context-free Grammar

- Represents the language using an ordered rooted tree.

- Root represents the starting symbol.
- Internal vertices represent the nonterminal symbol that arise in the production.
- Leaves represent the terminal symbols.

- If the production $A \to w$ arise in the derivation, where w is a word, the vertex that represents A has as children vertices that represent each symbol in w, in order from left to right.

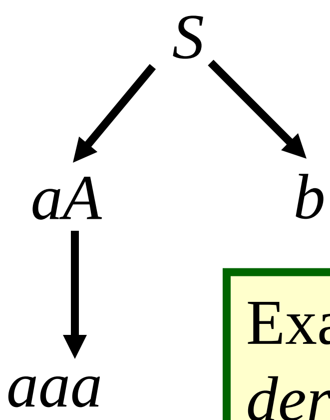
Language Generated by a Grammar

Example: Let $G = (\{S, A, a, b\}, \{a, b\}, S, \{S \rightarrow aA, S \rightarrow b, A \rightarrow aa\})$. What is $L(G)$?

Easy: We can just draw a tree of all possible derivations.

- We have: $S \Rightarrow aA \Rightarrow aaa$.
- and $S \Rightarrow b$.

Answer: $L = \{aaa, b\}$.



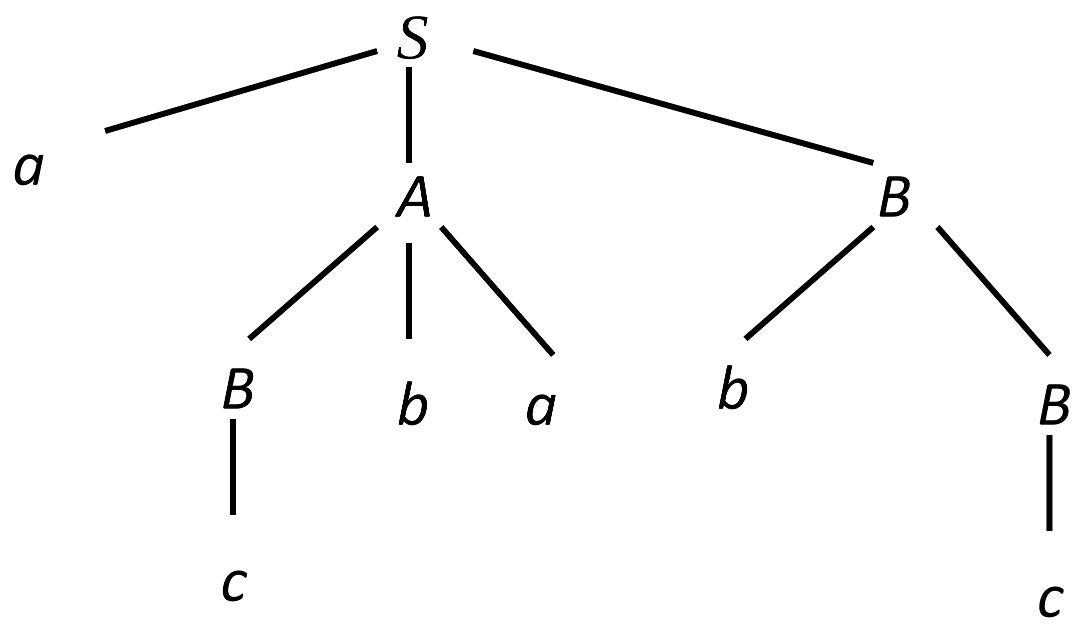
Example of a *derivation tree* or *parse tree* or *sentence diagram*.

Example: Derivation Tree

- Let G be a context-free grammar with the productions $P = \{S \to aAB, A \to Bba, B \to bB, B \to c\}$. The word w = acbabc can be derived from S as follows:

$$S \Rightarrow aAB \to a(Bba)B \Rightarrow acbaB \Rightarrow acba(bB) \Rightarrow acbabc$$

Thus, the derivation tree is given as follows:



Backus-Naur Form

⟨sentence⟩ ::= ⟨noun phrase⟩ ⟨verb phrase⟩

⟨noun phrase⟩ ::= ⟨article⟩ [⟨adjective⟩] ⟨noun⟩

⟨verb phrase⟩ ::= ⟨verb⟩ [⟨adverb⟩]

⟨article⟩ ::= a | the

⟨adjective⟩ ::= large | hungry

⟨noun⟩ ::= rabbit | mathematician

⟨verb⟩ ::= eats | hops

⟨adverb⟩ ::= quickly | wildly

Square brackets [] mean “optional”

Vertical bars mean “alternatives”

Generating Infinite Languages

A simple PSG can easily generate an infinite language.

Example: $S \rightarrow 11S$, $S \rightarrow 0$ ($T = \{0,1\}$).

The derivations are:

- $S \Rightarrow 0$
- $S \Rightarrow 11S \Rightarrow 110$
- $S \Rightarrow 11S \Rightarrow 1111S \Rightarrow 11110$
- and so on…

$L = \{(11)^*0\}$ – the set of all strings consisting of some number of concaten-ations of 11 with itself, followed by 0.

Another example

Construct a PSG that generates the language $L = \{\mathbf{0}^n\mathbf{1}^n \mid n \in \mathbf{N}\}$.

- $\mathbf{0}$ and $\mathbf{1}$ here represent symbols being concatenated n times, not integers being raised to the nth power.

Solution strategy: Each step of the derivation should preserve the invariant that the number of **0**'s = the number of **1**'s in the template so far, and all **0**'s come before all **1**'s.

Solution: $S \rightarrow \mathbf{0}S\mathbf{1}$, $S \rightarrow \lambda$.

Context-Sensitive Languages

The language $\{ a^n b^n c^n \mid n \geq 1\}$ is context-sensitive but not context free.

A grammar for this language is given by:

$$S \rightarrow aSBC \mid aBC$$

$$CB \rightarrow BC$$

$$aB \rightarrow ab$$

$$bB \rightarrow bb$$

$$bC \rightarrow bc$$

$$cC \rightarrow cc$$

Terminal
and
non-terminal

A derivation from this grammar is:-

$S \Rightarrow aSBC$

$\Rightarrow aaBCBC$ (using $S \rightarrow aBC$)

$\Rightarrow aabCBC$ (using $aB \rightarrow ab$)

$\Rightarrow aabBCC$ (using $CB \rightarrow BC$)

$\Rightarrow aabbCC$ (using $bB \rightarrow bb$)

$\Rightarrow aabbcC$ (using $bC \rightarrow bc$)

$\Rightarrow aabbcc$ (using $cC \rightarrow cc$)

which derives $a^2b^2c^2$.