Formal Languages applied to Linguistics

Pascal Amsili

Laboratoire de Linguistique Formelle, Université Paris Diderot

U. São Carlos, september 2014
Alphabet, word

Def. 1 (Alphabet)

An *alphabet* $\Sigma$ is a finite set of symbols (letters). The *size* of the alphabet is the cardinal of the set.

Def. 2 (Word)

A *word* on the alphabet $\Sigma$ is a finite sequence of letters from $\Sigma$. Formally, let $[p] = (1, 2, 3, 4, \ldots, p)$ (ordered integer sequence). Then a word is a mapping

$$u : [p] \rightarrow \Sigma$$

$p$, the length of $u$, is noted $|u|$. 
Examples I

Alphabet \{0,1,2,3,4,5,6,7,8,9, \cdot \} 
Words 235 \cdot 29  
007 \cdot 12  
\cdot 1 \cdot 1 \cdot 00 \cdots  
3.1415962\ldots (\pi)  
\cdots  

Alphabet \{\cdot, \square\}  
Words \square \square \square \cdot \cdot \cdot  
\cdot  
\cdot \cdot \cdot  
\cdots
Examples II

Alphabet \{ \_ , \_\_ , \_\_\_ , \_\_\_\_ , \_\_\_\_\_ , \_\_\_\_\_\_ , \ldots \} \\
Words \_ \_ \_ \_ \_ \_ \_ \_ \\
\_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \\
\_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \\
\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_ \\
\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \\
\ldots

Alphabet \{ a, \text{ man}, \text{ loves}, \text{ woman} \} \\
Words a \\
a \text{ man} \text{ loves a woman} \\
\text{ man} \text{ man a loves woman} \text{ loves a} \\
\ldots
Def. 3 ($\Sigma^*$)

Let $\Sigma$ be an alphabet. The set of all the words that can be formed with any number of letters from $\Sigma$ is noted $\Sigma^*$.

It comprises a word with no letter, noted $\varepsilon$.

Example: $\Sigma = \{a, b, c\}$

$\Sigma^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, \ldots, bbb, \ldots\}$

N.B.: $\Sigma^*$ is always infinite, except...
Monoid

Def. 3 (\( \Sigma^* \))

Let \( \Sigma \) be an alphabet. The set of all the words that can be formed with any number of letters from \( \Sigma \) is noted \( \Sigma^* \).

It comprises a word with no letter, noted \( \varepsilon \).

Example: \( \Sigma = \{a, b, c\} \)

\[
\Sigma^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, \ldots, bbb, \ldots\}
\]

N.B.: \( \Sigma^* \) is always infinite, except...

if \( \Sigma = \emptyset \). Then \( \Sigma^* = \{\varepsilon\} \).
Let $k$ be the size of the alphabet $k = |\Sigma|$.

Then $\Sigma^*$ contains:

- $k^0 = 1$ word(s) of 0 letters ($\varepsilon$)
- $k^1 = k$ word(s) of 1 letters
- $k^2$ word(s) of 2 letters
- \ldots
- $k^n$ words of $n$ letters, $\forall n \geq 0$
\[ \Sigma = \{a, b, c\} \]

- Words can be enumerated according to different orders
- \( \Sigma^* \) is a *countable* set
\( \Sigma^* \) can be equipped with a binary operation: the *concatenation*.

**Def. 4 (Concatenation)**

Let \([p] \xrightarrow{u} X\), \([q] \xrightarrow{w} X\). The concatenation of \(u\) and \(w\), noted \(uw\) (\(u.w\)) is thus defined:

\[
\begin{align*}
    uw &: [p + q] \rightarrow X \\
    uw_i &= \begin{cases} 
        u_i & \text{for } i \in [1, p] \\
        w_{i-p} & \text{for } i \in [p + 1, p + q]
    \end{cases}
\end{align*}
\]

Example:

\[
u = babca, \quad v = cca, \quad uv = babca cca\]
\[ \Sigma^* \text{ can be equipped with a binary operation: the } \text{concatenation} \]

**Def. 4 (Concatenation)**

Let \([p] \xrightarrow{u} X, [q] \xrightarrow{w} X\). The concatenation of \(u\) and \(w\), noted \(uw\) (\(u.w\)) is thus defined:

\[
\begin{align*}
    uw : & \quad [p + q] \rightarrow X \\
    uw_i &= \begin{cases} 
        u_i & \text{for } i \in [1, p] \\
        w_{i-p} & \text{for } i \in [p + 1, p + q]
    \end{cases}
\end{align*}
\]

Example : \( u \quad \text{bacba} \)  
\( v \quad \text{cca} \)
Concatenation

\[ \Sigma^* \] can be equipped with a binary operation: the *concatenation*

**Def. 4 (Concatenation)**

Let \([p] \xrightarrow{u} X, [q] \xrightarrow{w} X\). The concatenation of \(u\) and \(w\), noted \(uw\) (\(u.w\)) is thus defined:

\[
uw : [p + q] \longrightarrow X
\]

\[
uw_i = \begin{cases} 
u_i & \text{for } i \in [1, p] \\
w_{i-p} & \text{for } i \in [p + 1, p + q] \end{cases}
\]

Example:
\[
u = \text{bacba}
\]
\[
v = \text{cca}
\]
\[
uvw = \text{bacbaccba}
\]
Def. 5 (Factor)

A factor $w$ of $u$ is a subset of adjacent letters in $u$.

- $w$ is a factor of $u$ \iff \exists u_1, u_2 \text{ s.t. } u = u_1 w u_2$
- $w$ is a left factor (prefix) of $u$ \iff \exists u_2 \text{ s.t. } u = w u_2$
- $w$ is a right factor (suffix) of $u$ \iff \exists u_1 \text{ s.t. } u = u_1 w$

Def. 6 (Factorization)

We call factorization the decomposition of a word in factors.
Role of concatenation

1. Words have been defined on $\Sigma$. If one takes two such words, it’s always possible to form a new word by concatenating them.

2. Any word can be factorised in many different ways:

   $abaccab$
Role of concatenation

1. Words have been defined on $\Sigma$. If one takes two such words, it’s always possible to form a new word by concatenating them.

2. Any word can be factorised in many different ways:
   $$abaccab$$
   $$((ab)(a)(c)(c)(a)(b))$$
Role of concatenation

1. Words have been defined on $\Sigma$. If one takes two such words, it’s always possible to form a new word by concatenating them.

2. Any word can be factorised in many different ways:

$$abaccab$$

$$(ab)(acc)(ab)$$
Role of concatenation

1. Words have been defined on $\Sigma$. If one takes two such words, it’s always possible to form a new word by concatenating them.

2. Any word can be factorised in many different ways:
   
   $abaccab$
   
   $((((((ab)a)c)c)a)b))$
Role of concatenation

1. Words have been defined on $\Sigma$. If one takes two such words, it’s always possible to form a new word by concatenating them.

2. Any word can be factorised in many different ways:

$abacccab$

$(a)(b)(a)(c)(c)(a)(b)$
Role of concatenation

1. Words have been defined on $\Sigma$. If one takes two such words, it’s always possible to form a new word by concatenating them.

2. Any word can be factorised in many different ways:
   
   $abaccab$
   
   $((ab)(ac)(ca)(ba))$

3. Since all letters of $\Sigma$ form a word of length 1 (this set of words is called the base),

4. any word of $\Sigma^*$ can be seen as a (unique) sequence of concatenations of length 1 words:

   $abaccab$
   
   $((((((ab)a)c)c)a)b)$
   
Properties of concatenation

1. Concatenation is non commutative
2. Concatenation is associative
3. Concatenation has an identity (neutral) element: \( \varepsilon \)

Notation: \( a.a.a = a^3 \)

1. \( uv.w \neq w.uv \)
2. \( (u.v).w = u.(v.w) \)
3. \( u.\varepsilon = \varepsilon.u = u \)
Overview

1. Formal Languages
   - Base notions
   - Definition
   - Problem

2. Formal Grammars

3. Regular Languages

4. Formal complexity of Natural Languages
Def. 7 ((Formal) Language)

Let $\Sigma$ be an alphabet.
A language on $\Sigma$ is a set of words on $\Sigma$. 

Def. 7 ((Formal) Language)

Let $\Sigma$ be an alphabet.

A language on $\Sigma$ is a set of words on $\Sigma$.

or, equivalently,

A language on $\Sigma$ is a subset of $\Sigma^*$
Examples I

Let $\Sigma = \{a, b, c\}$. 
Examples 1

Let $\Sigma = \{a, b, c\}$.

$L_1 = \{aa, ab, bac\}$ finite language
Examples I

Let $\Sigma = \{a, b, c\}$.

$L_1 = \{aa, ab, bac\}$ finite language

$L_2 = \{a, aa, aaa, aaaa \ldots\}$
Let $\Sigma = \{a, b, c\}$.

$L_1 = \{aa, ab, bac\}$ \hspace{1cm} \text{finite language}

$L_2 = \{a, aa, aaa, aaaa \ldots\}$

or $L_2 = \{a^i / i \geq 1\}$ \hspace{1cm} \text{infinite language}
Examples I

Let $\Sigma = \{a, b, c\}$.

\[
\begin{array}{ll}
L_1 = \{aa, ab, bac\} & \text{finite language} \\
L_2 = \{a, aa, aaa, aaaa \ldots\} & \text{infinite language} \\
\quad \text{or } L_2 = \{a^i \mid i \geq 1\} & \text{infinite language} \\
L_3 = \{\varepsilon\} & \text{finite language, reduced to a singleton}
\end{array}
\]
**Examples I**

Let $\Sigma = \{a, b, c\}$.

<table>
<thead>
<tr>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$</td>
<td>${aa, ab, bac}$ finite language</td>
</tr>
<tr>
<td>$L_2$</td>
<td>${a, aa, aaa, aaaa \ldots}$ infinite language</td>
</tr>
<tr>
<td></td>
<td>or $L_2 = {a^i / i \geq 1}$ infinite language</td>
</tr>
<tr>
<td>$L_3$</td>
<td>${\varepsilon}$ finite language reduced to a singleton</td>
</tr>
<tr>
<td></td>
<td>$\neq$</td>
</tr>
</tbody>
</table>
Let $\Sigma = \{a, b, c\}$.

<table>
<thead>
<tr>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1 = {aa, ab, bac}$</td>
<td>finite language</td>
</tr>
<tr>
<td>$L_2 = {a, aa, aaa, aaaa \ldots}$</td>
<td>infinite language</td>
</tr>
<tr>
<td>or $L_2 = {a^i / i \geq 1}$</td>
<td></td>
</tr>
<tr>
<td>$L_3 = {\varepsilon}$</td>
<td>finite language, reduced to a singleton</td>
</tr>
<tr>
<td>$L_4 = \emptyset$</td>
<td>“empty” language</td>
</tr>
</tbody>
</table>
Examples I

Let $\Sigma = \{a, b, c\}$.

<table>
<thead>
<tr>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1 = {aa, ab, bac}$</td>
<td>finite language</td>
</tr>
<tr>
<td>$L_2 = {a, aa, aaa, aaaa \ldots}$</td>
<td>infinite language</td>
</tr>
<tr>
<td>or $L_2 = {a^i / i \geq 1}$</td>
<td></td>
</tr>
<tr>
<td>$L_3 = {\varepsilon}$</td>
<td>finite language, reduced to a singleton</td>
</tr>
<tr>
<td>$L_4 = \emptyset$</td>
<td>“empty” language</td>
</tr>
<tr>
<td>$L_5 = \Sigma^*$</td>
<td></td>
</tr>
</tbody>
</table>
Examples II

Let $\Sigma = \{a, \text{man}, \text{loves}, \text{woman}\}$. 
Examples II

Let $\Sigma = \{a, \text{man, loves, woman}\}$.

$L = \{ \text{a man loves a woman, a woman loves a man} \}$
Examples II

Let $\Sigma = \{a,\ man,\ loves,\ woman\}$.

$L = \{\ a\ man\ loves\ a\ woman,\ a\ woman\ loves\ a\ man\ \}$

Let $\Sigma' = \{a,\ man,\ who,\ saw,\ fell\}$. 
Examples II

Let $\Sigma = \{ \text{a, man, loves, woman} \}$. 

$L = \{ \text{a man loves a woman, a woman loves a man } \}$

Let $\Sigma' = \{ \text{a, man, who, saw, fell} \}$. 

$L' = \left\{ \begin{array}{l}
\text{a man fell,} \\
\text{a man who saw a man fell,} \\
\text{a man who saw a man who saw a man fell,} \\
\ldots
\end{array} \right\}$
Set operations

Since a language is a set, usual set operations can be defined:

- union
- intersection
- set difference
Set operations

Since a language is a set, usual set operations can be defined:

- union
- intersection
- set difference

⇒ One may describe a (complex) language as the result of set operations on (simpler) languages:
\[
\{a^{2k} / k \geq 1\} = \{a, aa, aaa, aaaa, \ldots\} \cap \{ww / w \in \Sigma^*\}
\]
Def. 8 (product operation on languages)

One can define the language product and its closure the Kleene star operation:

- The **product** of languages is thus defined:
  \[ L_1 \cdot L_2 = \{ uv / u \in L_1 \land v \in L_2 \} \]

  Notation: \( L \cdot L \cdot L \ldots \cdot L = L^k ; L^0 = \{ \varepsilon \} \)

- The Kleene star of a language is thus defined:
  \[ L^* = \bigcup_{n \geq 0} L^n \]
It is common to use the 3 *rational* operations:

- union
- product
- Kleene star

To characterize certain languages...
Regular expressions

It is common to use the 3 *rational* operations:

- union
- product
- Kleene star

...to characterize certain languages...

\[(\{a\} \cup \{b\})^*\cdot \{c\} = \{c, ac, abc, bc, \ldots, baabaac, \ldots\}\]

(simplified notation \((a|b)^*c\) — regular expressions)
Regular expressions

It is common to use the 3 *rational* operations:

- union
- product
- Kleene star

to characterize certain languages...

\[(\{a\} \cup \{b\})^* \cdot \{c\} = \{c, ac, abc, bc, \ldots, baabaac, \ldots\}\]

(simplified notation \((a|b)^*c\) — regular expressions)

... but not all languages can be thus characterized.
Overview

1. Formal Languages
   - Base notions
   - Definition
   - Problem

2. Formal Grammars

3. Regular Languages

4. Formal complexity of Natural Languages
Back to “Natural” Languages

English as a formal language:

- **alphabet** morphemes (often simplified to words —depending on your view on flexional morphology)
  \[\Rightarrow\] Finite at a time \( t \) by hypothesis

- **words** well formed English sentences
  \[\Rightarrow\] English sentences are all finite by hypothesis

- **language** English, as a set of an infinite number of well formed combinations of “letters” from the alphabet
Discussion I

1. is the alphabet finite?

   closed class morphemes obviously
   open class morphemes what about “new words”?

   morphological derivations can be seen as
   produced from an unchanged inventory (1)

   other words
   - loan words (rare)
   - lexical inventions (rare)
   - change of category (2) (bounded)

   $\Rightarrow$ negligible

(1) motherese = mother+ese

(2) $\text{american}_A \rightarrow \text{american}_N$
Discussion II

2. is English infinite?
   - It is supposed that you can always produce a longer sentence than the previous one by adding linguistic material preserving well-formedness.
   - Compatible with the working memory limit
     (Langendoen & Postal, 1984)

3. is language discrete?
   Well, that's another story
Linguists sometimes have trouble with infinity:
In order for there to be an infinite number of sentences in a language there must either be an infinite number of words in the language (clearly not true) or there must be the possibility of infinite length sentences. The product of two finite numbers is always a finite number. (Mannell, 1999)

and many others
Linguists sometimes have trouble with infinity:

In order for there to be an infinite number of sentences in a language there must either be an infinite number of words in the language (clearly not true) or there must be the possibility of infinite length sentences. The product of two finite numbers is always a finite number.

(Mannell, 1999)

and many others

!! WRONG !!
Linguists sometimes have trouble with infinity:

In order for there to be an infinite number of sentences in a language there must either be an infinite number of words in the language (clearly not true) or there must be the possibility of infinite length sentences. The product of two finite numbers is always a finite number. (Mannell, 1999)

and many others

!! WRONG !!

The whole point of formal languages is that they are infinite sets of finite words on a finite alphabet.
Linguists sometimes have trouble with infinity:

In order for there to be an infinite number of sentences in a language there must either be an infinite number of words in the language (clearly not true) or there must be the possibility of infinite length sentences. The product of two finite numbers is always a finite number. (Mannell, 1999) and many others

!! WRONG !!

The whole point of formal languages is that they are infinite sets of finite words on a finite alphabet.

von Humbolt: language is an infinite use of finite means (quoted by Chomsky)
Good questions

Why would one consider natural language as a formal language?

- it allows to describe the language in a formal/compact/elegant way
- it allows to compare various languages (via classes of languages established by mathematicians)
- it gives algorithmic tools to recognize and to analyse words of a language.

\[ \text{recognize } u : \text{ decide whether } u \in L \]
\[ \text{analyse } u : \text{ show the internal structure of } u \]
Overview

1. Formal Languages

2. Formal Grammars
   - Definition
   - Language classes

3. Regular Languages

4. Formal complexity of Natural Languages
Formal grammars have been proposed by Chomsky as one of the available means to characterize a formal language. Other means include:

- Turing machines (automata)
- $\lambda$-terms
- ...
Def. 9 ((Formal) Grammar)

A formal grammar is defined by \( \langle \Sigma, N, S, P \rangle \) where

- \( \Sigma \) is an alphabet
- \( N \) is a disjoint alphabet non-terminal vocabulary
- \( S \in V \) is a distinguished element of \( N \), called the axiom
- \( P \) is a set of « production rules », namely a subset of the cartesian product \((\Sigma \cup N)^* N (\Sigma \cup N)^* \times (\Sigma \cup N)^*\).
Examples

\[ G_0 = \langle \Sigma, N, S, P \rangle \]
Examples

\[ \langle \Sigma, N, S, P \rangle \]

\[ G_0 = \langle \{ \text{joe}, \text{sam}, \text{sleeps} \} \rangle, \]
Examples

\[ \langle \Sigma, N, S, P \rangle \]

\[ G_0 = \left\langle \{\text{joe, sam, sleeps}\}, \{N, V, S\} , \right\rangle \]
Examples

\[ \langle \Sigma, N, S, P \rangle \]

\[ G_0 = \left( \{ \text{joe, sam, sleeps} \}, \{ N, V, S \}, S, \right) \]
Examples

\[ G_0 = \langle \Sigma, N, S, P \rangle \]

\[
\begin{align*}
G_0 &= \left\langle \{joe, sam, sleeps\}, \{N, V, S\}, S, \right. \\
&\quad \left. \left\{ (N, joe), (N, sam), (V, sleeps), (S, N V) \right\} \right\rangle
\end{align*}
\]
Examples

\[ G_0 = \langle \Sigma, N, S, P \rangle \]

\[ G_0 = \left\{ \{\text{joe, sam, sleeps}\}, \{N, V, S\}, S, \begin{cases} N \rightarrow \text{joe} \\ N \rightarrow \text{sam} \\ V \rightarrow \text{sleeps} \\ S \rightarrow N \ V \end{cases} \right\} \]
Examples (cont’d)

\[ G_1 = \left\{ \{\text{jean, dort}\}, \{\text{Np, SN, SV, V, S}\}, S, \right\} \]

\[ G_2 = \langle \{(,),\}\}, \{S\}, S, \{S \rightarrow \varepsilon | (S)S\} \rangle \]

\[ \begin{align*}
S & \rightarrow SN \; SV \\
SN & \rightarrow Np \\
SV & \rightarrow V \\
Np & \rightarrow \text{jean} \\
V & \rightarrow \text{dort}
\end{align*} \]
\[ G_3 : \]
\[
E \rightarrow E + E \\
| \quad E \times E \\
| \quad (E) \\
| \quad F \\
F \rightarrow 0\,|\,1\,|\,2\,|\,3\,|\,4\,|\,5\,|\,6\,|\,7\,|\,8\,|\,9
\]
Formal Languages
Formal Grammars
Regular Languages
Formal complexity of Natural Languages
References

Notation

\[ G_3 : \quad E \quad \rightarrow \quad E + E \]
| \quad E \times E |
| \quad (E) |
| \quad F |

\[ F \quad \rightarrow \quad 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 \]

\[ G_3 = \langle \{ +, \times, (, ), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}, \{ E, F \}, E, \{ \ldots \} \rangle \]
Formal Languages
Formal Grammars
Regular Languages
Formal complexity of Natural Languages
References

Notation

\[ G_3 : \]
\[ E \rightarrow E + E \]
\[ | \quad E \times E \]
\[ | \quad (E) \]
\[ | \quad F \]
\[ F \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \]

\[ G_3 = \langle \{+, \times, (,), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \{E, F\}, E, \{\ldots\} \rangle \]

\[ G_4 = E \rightarrow E + T \mid T \]
\[ T, T \rightarrow T \times F \mid F, F \rightarrow (E) \mid a \]
Immediate Derivation

**Def. 10 (Immediate derivation)**

Let \( G = \langle X, V, S, P \rangle \) a grammar, \((f, g) \in (X \cup V)^*\) two “words”, \(r \in P\) a production rule, such that \( r : A \rightarrow u \ (u \in (X \cup V)^*)\).

- \( f \) derives into \( g \) (immediate derivation) with the rule \( r \) (noted \( f \xrightarrow{r} g \)) iff
  \[ \exists v, w \text{ s.t. } f = vAuw \text{ and } g = vuw \]

- \( f \) derives into \( g \) (immediate derivation) in the grammar \( G \) (noted \( f \xrightarrow{G} g \)) iff
  \[ \exists r \in P \text{ s.t. } f \xrightarrow{r} g. \]
Def. 11 (Derivation)

\[ f \xrightarrow{G^*} g \text{ if } f = g \]
\[ \exists f_0, f_1, f_2, \ldots, f_n \text{ s.t. } f_0 = f \]
\[ f_n = g \]
\[ \forall i \in [1, n] : f_{i-1} \xrightarrow{g} f_i \]

An example with \( G_0 \):
\[ N \ V \ joe \ N \]
Definition 11 (Derivation)

\[ f \xrightarrow{G^*} g \text{ if } f = g \]

\[ \exists f_0, f_1, f_2, \ldots, f_n \text{ s.t. } f_0 = f \]
\[ f_n = g \]
\[ \forall i \in [1, n] : f_{i-1} \xrightarrow{G} f_i \]

An example with \( G_0 \):

\[ N \ V \ joe \ N \rightarrow sam \ V \ joe \ N \]
Def. 11 (Derivation)

\[ f \xrightarrow{G^*} g \text{ if } f = g \]

\[ \exists f_0, f_1, f_2, \ldots, f_n \text{ s.t. } f_0 = f \]
\[ f_n = g \]
\[ \forall i \in [1, n] : f_{i-1} \xrightarrow{g} f_i \]

or

An example with \( G_0 \):

\[ NV\,joe\,N \rightarrow sam\,V\,joe\,N \rightarrow sam\,V\,joe\,joe \]
Def. 11 (Derivation)

\[ f \xrightarrow{G^*} g \text{ if } f = g \]

\[ \exists f_0, f_1, f_2, \ldots, f_n \text{ s.t. } f_0 = f \]

\[ f_n = g \]

\[ \forall i \in [1, n] : f_{i-1} \xrightarrow{G} f_i \]

An example with \( G_0 \):

\[ N \ V \ joe \ N \rightarrow sam \ V \ joe \ N \rightarrow sam \ V \ joe \ joe \]

or

\[ sam \ V \ joe \ sam \]
Def. 11 (Derivation)

\[ f \xrightarrow{G^*} g \text{ if } f = g \]

\[ \exists f_0, f_1, f_2, \ldots, f_n \text{ s.t. } f_0 = f, f_n = g, \forall i \in [1, n] : f_{i-1} \xrightarrow{g} f_i \]

An example with \( G_0 \):

\[ N \ V \ joe \ N \xrightarrow{} sam \ V \ joe \ N \xrightarrow{} sam \ V \ joe \ joe \]

\[ \text{ or } sam \ V \ joe \ sam \]

\[ \text{ or } sam \ sleeps \ joe \ N \]

\[ \ldots \]
Endpoint of a derivation

\[ G_3 : \ E \quad \rightarrow \quad E + E \]

\[ \quad | \quad E \times E \]

\[ \quad | \quad (E) \]

\[ \quad | \quad F \]

\[ F \quad \rightarrow \quad 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \]

An example with \( G_3 \):

\[ E \times E \]
Endpoint of a derivation

\[ G_3 : \quad E \quad \rightarrow \quad E + E \]
\[ \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \qua
Endpoint of a derivation

\[ G_3 : \quad \begin{array}{c}
E \rightarrow E + E \\
| \\
E \times E \\
| \\
( E ) \\
| \\
F \\
F \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array} \]

An example with \( G_3 \):

\[ E \times E \rightarrow F \times E \rightarrow 3 \times E \]
Endpoint of a derivation

\[ G_3 : \quad E \rightarrow E + E \\
\quad \quad | \quad E \times E \\
\quad \quad | \quad (E) \\
\quad \quad | \quad F \\
F \rightarrow 0 \cdot 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \]

An example with \( G_3 \):

\[ E \times E \rightarrow F \times E \rightarrow 3 \times E \rightarrow 3 \times (E) \]
Endpoint of a derivation

\[ G_3 : \quad E \rightarrow E + E \]
\[ \quad \quad | \quad E \times E \]
\[ \quad \quad | \quad (E) \]
\[ \quad \quad | \quad F \]
\[ F \rightarrow 0|1|2|3|4|5|6|7|8|9 \]

An example with \( G_3 \):

\[ E \times E \rightarrow F \times E \rightarrow 3 \times E \rightarrow 3 \times (E) \rightarrow 3 \times (E + E) \]
Endpoint of a derivation

\[ G_3 : \]

\[
E \rightarrow E + E \\
E \times E \\
(E) \\
F
\]

\[
F \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
\]

An example with \( G_3 \):

\[
E \times E \rightarrow F \times E \rightarrow 3 \times E \rightarrow 3 \times (E) \rightarrow 3 \times (E + E) \rightarrow 3 \times (E + F)
\]
Endpoint of a derivation

\[ G_3 : \begin{align*}
E & \rightarrow E + E \\
| & E \times E \\
| & (E) \\
| & F \\
F & \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
\end{align*} \]

An example with \( G_3 \):

\[ E \times E \rightarrow F \times E \rightarrow 3 \times E \rightarrow 3 \times (E) \rightarrow 3 \times (E + E) \rightarrow 3 \times (E + F) \rightarrow 3 \times (E + 4) \]
Endpoint of a derivation

$G_3 : \ E \rightarrow E + E$

|   | $E \times E$
|---|---
|   | $(E)$
|   | $F$

$F \rightarrow 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9$

An example with $G_3$:

$E \times E \rightarrow F \times E \rightarrow 3 \times E \rightarrow 3 \times (E) \rightarrow 3 \times (E + E) \rightarrow$

$3 \times (E + F) \rightarrow 3 \times (E + 4) \rightarrow 3 \times (F + 4)$
Endpoint of a derivation

\[ G_3 : \ E \rightarrow E + E \]
| \[ E \times E \]
| \[ (E) \]
| \[ F \]

\[ F \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 \]

An example with \( G_3 \):

\[ E \times E \rightarrow F \times E \rightarrow 3 \times E \rightarrow 3 \times (E) \rightarrow 3 \times (E + E) \rightarrow 3 \times (E + F) \rightarrow 3 \times (E + 4) \rightarrow 3 \times (F + 4) \rightarrow 3 \times (5 + 4) \]
Endpoint of a derivation

\[ G_3 : \begin{align*}
E & \rightarrow E + E \\
| & \rightarrow E \times E \\
| & \rightarrow (E) \\
| & \rightarrow F \\
F & \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
\]

An example with \( G_3 \):

\[ E \times E \rightarrow F \times E \rightarrow 3 \times E \rightarrow 3 \times (E) \rightarrow 3 \times (E + E) \rightarrow 3 \times (E + F) \rightarrow 3 \times (E + 4) \rightarrow 3 \times (F + 4) \rightarrow 3 \times (5 + 4) \rightarrow \]
Engendered language

Def. 12 (Language engendered by a word)

Let \( f \in (\Sigma \cup N)^* \).

\[ L_G(f) = \{ g \in X^* / f \xrightarrow{G*} g \} \]

Def. 13 (Language engendered by a grammar)

The language engendered by a grammar \( G \) is the set of words of \( \Sigma^* \) derived from the axiom.

\[ L_G = L_G(S) \]
Engendered language

Def. 12 (Language engendered by a word)
Let $f \in (\Sigma \cup N)^*$. 
$L_G(f) = \{g \in X^* / f \xrightarrow{G}\! g\}$

Def. 13 (Language engendered by a grammar)
The language engendered by a grammar $G$ is the set of words of $\Sigma^*$ derived from the axiom.
$L_G = L_G(S)$

For instance $() \in L_{G_2}$:
Engendered language

Def. 12 (Language engendered by a word)

Let $f \in (\Sigma \cup N)^*$.  
$L_G(f) = \{ g \in X^* / f \xrightarrow{G^*} g \}$

Def. 13 (Language engendered by a grammar)

The language engendered by a grammar $G$ is the set of words of $\Sigma^*$ derived from the axiom.
$L_G = L_G(S)$

For instance $(,) \in L_{G_2}: S \rightarrow (S)S$
Engendered language

Def. 12 (Language engendered by a word)
Let \( f \in (\Sigma \cup N)^* \).
\[
L_G(f) = \{ g \in X^*/ f \xrightarrow{G^*} g \}
\]

Def. 13 (Language engendered by a grammar)
The language engendered by a grammar \( G \) is the set of words of \( \Sigma^* \) derived from the axiom.
\[
L_G = L_G(S)
\]

For instance \((() \in L_{G_2}: S \rightarrow (S)S \rightarrow ()S)\).
Engendered language

**Def. 12 (Language engendered by a word)**

Let \( f \in (\Sigma \cup N)^* \).

\[ L_G(f) = \{ g \in X^*/f \xrightarrow{G^*} g \} \]

**Def. 13 (Language engendered by a grammar)**

The *language engendered by a grammar* \( G \) is the set of words of \( \Sigma^* \)
derived from the *axiom*.

\[ L_G = L_G(S) \]

For instance \((\) \( ) \in L_G_2: S \rightarrow (S)S \rightarrow ( )S \rightarrow ( )\)
Engendered language

Def. 12 (Language engendered by a word)

Let $f \in (\Sigma \cup N)^*$. 
$L_G(f) = \{ g \in X^* / f \xrightarrow{G^*} g \}$

Def. 13 (Language engendered by a grammar)

The language engendered by a grammar $G$ is the set of words of $\Sigma^*$ derived from the axiom. 
$L_G = L_G(S)$

For instance $(\) \in L_G_2: S \rightarrow (S)S \rightarrow (\)S \rightarrow (\)$
as well as $((())), (())(), (((()))))\ldots$
Engendered language

Def. 12 (Language engendered by a word)
Let $f \in (\Sigma \cup N)^*$. 
$L_G(f) = \{g \in X^*/f \xrightarrow{G^*} g\}$

Def. 13 (Language engendered by a grammar)
The language engendered by a grammar $G$ is the set of words of $\Sigma^*$ derived from the axiom. 
$L_G = L_G(S)$

For instance $(\in L_G_2: S \rightarrow (S)S \rightarrow (S)S \rightarrow ()$ 
as well as $((()))$, $())()$, $((())())$... 
but $))( \notin L_G_2$, even though the following is a licit derivation:
Engendered language

**Def. 12 (Language engendered by a word)**

Let \( f \in (\Sigma \cup N)^* \).

\[ L_G(f) = \{ g \in X^* / f \xrightarrow{G^*} g \} \]

**Def. 13 (Language engendered by a grammar)**

The *language engendered by a grammar* \( G \) is the set of words of \( \Sigma^* \) derived from the *axiom*.

\[ L_G = L_G(S) \]

For instance \( () \in L_{G_2} \): \( S \rightarrow (S)S \rightarrow ()S \rightarrow () \)

as well as \( (((()))), ()(), (((()))()) \ldots \)

but \( )()() \notin L_{G_2} \), even though the following is a licit derivation:

\( )S( \rightarrow \)
Engendered language

Def. 12 (Language engendered by a word)
Let $f \in (\Sigma \cup N)^*$.
$L_G(f) = \{ g \in X^* / f \xrightarrow{G^*} g \}$

Def. 13 (Language engendered by a grammar)
The *language engendered by a grammar* $G$ is the set of words of $\Sigma^*$ derived from the *axiom*.
$L_G = L_G(S)$

For instance $(\in L_G_2: S \rightarrow (S)S \rightarrow ()S \rightarrow ()$
as well as $(())), (())(), (((())()))...$
buts $(\notin L_G_2$, even though the following is a licit derivation:
$)S( \rightarrow )()S( \rightarrow$
Engendered language

Def. 12 (Language engendered by a word)

Let \( f \in (\Sigma \cup N)^* \).

\[ L_G(f) = \{ g \in X^* \mid f \xrightarrow{G^*} g \} \]

Def. 13 (Language engendered by a grammar)

The language engendered by a grammar \( G \) is the set of words of \( \Sigma^* \) derived from the axiom.

\[ L_G = L_G(S) \]

For instance \( () \in L_{G_2} : S \rightarrow (S)S \rightarrow ()S \rightarrow () \) as well as \( (((())) \), \( ()() () \), \( (((()))()()) \) \ldots \)

but \( )()() \not\in L_{G_2} \), even though the following is a licit derivation:

\[ )S( \rightarrow ))(S)S( \rightarrow )())S( \rightarrow \)
**Engendered language**

**Def. 12 (Language engendered by a word)**

Let \( f \in (\Sigma \cup \mathbb{N})^* \).

\[
L_G(f) = \{ g \in X^*/f \xrightarrow{G^*} g \}
\]

**Def. 13 (Language engendered by a grammar)**

The *language engendered by a grammar* \( G \) is the set of words of \( \Sigma^* \) derived from the *axiom*.

\[ L_G = L_G(S) \]

For instance \((\) \in L_{G_2}: S \rightarrow (S)S \rightarrow (S) \rightarrow (\) as well as \(((())\), \(()()\), \(((())())())\)\)... but \)()\( \not\in L_{G_2}, even though the following is a licit derivation:

\( )S( \rightarrow ))(S)S( \rightarrow ))()S( \rightarrow )))(\)
Engendered language

Def. 12 (Language engendered by a word)

Let \( f \in (\Sigma \cup N)^* \).

\[ L_G(f) = \{ g \in X^* / f \xrightarrow{G^*} g \} \]

Def. 13 (Language engendered by a grammar)

The \textit{language engendered by a grammar} \( G \) is the set of words of \( \Sigma^* \) derived from the \textit{axiom}.

\[ L_G = L_G(S) \]

For instance \((\)\) \(\in L_{G_2} : S \rightarrow (S)S \rightarrow (S) \rightarrow (\)

as well as \(((())())(), ()(), (((())())())\ldots\)

but \( )()() \notin L_{G_2} \), even though the following is a licit derivation:

\[ )S( \rightarrow ))(S)S( \rightarrow )())S( \rightarrow )())(\)

for there is no way to arrive at \( )S( \) starting with \( S \).
Example

\[ G_4 = E \to E + T \mid T, T \to T \times F \mid F, F \to (E) \mid a \]

\[ a + a, a + (a \times a), \ldots \]
Def. 14 (Proto-word)

A proto-word (or proto-sentence) is a word on \((\Sigma \cup N)^* \cdot N(\Sigma \cup N)^*\) (that is, a word containing at least one letter of \(N\)) produced by a derivation from the axiom.

\[
E \rightarrow E + T \rightarrow E + T \cdot F \rightarrow T + T \cdot F \rightarrow T + F \cdot F \rightarrow T + a \cdot F \rightarrow F + a \cdot F \rightarrow a + a \cdot F \rightarrow a\,a\,a\,a
\]
A given word may have several derivations:

\[ E \rightarrow E + E \rightarrow F + E \rightarrow F + F \rightarrow 3 + F \rightarrow 3 + 4 \]
Multiple derivations

A given word may have several derivations:

\[ E \rightarrow E + E \rightarrow F + E \rightarrow F + F \rightarrow 3 + F \rightarrow 3 + 4 \]
\[ E \rightarrow E + E \rightarrow E + F \rightarrow E + 4 \rightarrow F + 4 \rightarrow 3 + 4 \]
A given word may have several derivations:

\[ E \rightarrow E + E \rightarrow F + E \rightarrow F + F \rightarrow 3 + F \rightarrow 3 + 4 \]
\[ E \rightarrow E + E \rightarrow E + F \rightarrow E + 4 \rightarrow F + 4 \rightarrow 3 + 4 \]

... but if the grammar is not ambiguous, there is only one left derivation:
Multiple derivations

A given word may have several derivations:

\[ E \rightarrow E + E \rightarrow F + E \rightarrow F + F \rightarrow 3 + F \rightarrow 3 + 4 \]
\[ E \rightarrow E + E \rightarrow E + F \rightarrow E + 4 \rightarrow F + 4 \rightarrow 3 + 4 \]

... but if the grammar is not ambiguous, there is only one left derivation:

\[ E \rightarrow E + E \rightarrow E + E \rightarrow 3 + E \rightarrow 3 + F \rightarrow 3 + 4 \]
Multiple derivations

A given word may have several derivations:

\[ E \rightarrow E + E \rightarrow F + E \rightarrow F + F \rightarrow 3 + F \rightarrow 3 + 4 \]

\[ E \rightarrow E + E \rightarrow E + F \rightarrow E + 4 \rightarrow F + 4 \rightarrow 3 + 4 \]

... but if the grammar is not ambiguous, there is only one **left** derivation:

\[ E \rightarrow E + E \rightarrow F + E \rightarrow 3 + E \rightarrow 3 + F \rightarrow 3 + 4 \]

*Parsing*: trying to find the/a left derivation (resp. right)
For context-free languages, there is a way to represent the set of equivalent derivations, via a derivation tree which shows all the derivation independantly of their order.

Grammar $G_2$: $S \rightarrow \varepsilon$

\begin{align*}
S &\rightarrow (S)S \\
&\rightarrow ((S)S)S \\
&\rightarrow ((S)S) \\
&\rightarrow ((S)) \\
&\rightarrow ()
\end{align*}
Syntactic trees are precious to give access to the semantics
Ambiguity

When a grammar can assign more than one derivation tree to a word \( w \in L(G) \) (or more than one left derivation), the grammar is **ambiguous**.

For instance, \( G_3 \) is ambiguous, since it can assign the two following trees to \( 1 + 2 \times 3 \):

\[
\begin{align*}
E & \quad + \quad E \\
F & \quad E & \quad \times & \quad E \\
1 & \quad F & \quad 2 & \quad 3
\end{align*}
\]

\[
\begin{align*}
E & \quad \times \quad E \\
E & \quad + \quad E \\
F & \quad F & \quad 3 \\
1 & \quad 2
\end{align*}
\]
About ambiguity

- Ambiguity is not desirable for the semantics
- Useful artificial languages are rarely ambiguous
- There are context-free languages that are intrinsequely ambiguous (3)
- Natural languages are notoriously ambiguous...

(3) \[ \{ a^n ba^m ba^p ba^q | (n \geq q \land m \geq p) \lor (n \geq m \land p \geq q) \} \]
Comparison of grammars

- different languages generated $\Rightarrow$ different grammars
- same language generated by $G$ and $G'$:
  $\Rightarrow$ same weak generative power
- same language generated by $G$ and $G'$, and same structural decomposition:
  $\Rightarrow$ same strong generative power
Overview

1. Formal Languages
2. Formal Grammars
   - Definition
   - Language classes
3. Regular Languages
4. Formal complexity of Natural Languages
Define language families on the basis of properties of the grammars that generate them:

1. Four classes are defined, they are included one in another.
2. A language is of type $k$ if it can be recognized by a type $k$ grammar (and thus, by definition, by a type $k - 1$ grammar); and cannot be recognized by a grammar of type $k + 1$. 
Chomsky’s hierarchy

**type 0** No restriction on
\[ P \subseteq (X \cup V)^* V(X \cup V)^* \times (X \cup V)^*. \]

**type 1** (*context-sensitive* grammars) All rules of \( P \) are of the shape \((u_1 Su_2, u_1 mu_2)\), where \( u_1 \) and \( u_2 \in (X \cup V)^* \), \( S \in V \) and \( m \in (X \cup V)^+ \).

**type 2** (*context-free grammar*) All rules of \( P \) are of the shape \((S, m)\), where \( S \in V \) and \( m \in (X \cup V)^* \).

**type 3** (*regular grammars*) All rules of \( P \) are of the shape \((S, m)\), where \( S \in V \) and \( m \in X.V \cup X \cup \{\varepsilon\} \).
Examples

type 3:

\[
S \rightarrow aS | aB | bB | cA \\
B \rightarrow bB | b \\
A \rightarrow cS | bB
\]
Examples

type 3:
\[
S \rightarrow aS \mid aB \mid bB \mid cA \\
B \rightarrow bB \mid b \\
A \rightarrow cS \mid bB
\]

type 2:
\[
E \rightarrow E + T \mid T, T \rightarrow T \times F \mid F, F \rightarrow (E) \mid a
\]
Example 1 type 0

Type 0:

\[
\begin{align*}
S & \rightarrow SABC \\
S & \rightarrow \varepsilon \\
AC & \rightarrow CA \\
AB & \rightarrow BA \\
BA & \rightarrow AB \\
A & \rightarrow a \\
CA & \rightarrow AC \\
BC & \rightarrow CB \\
CB & \rightarrow BC \\
B & \rightarrow b \\
C & \rightarrow c
\end{align*}
\]

generated language:
Example 1 type 0

Type 0:

\[
\begin{align*}
S & \rightarrow SABC & AC & \rightarrow CA & A & \rightarrow a \\
S & \rightarrow \varepsilon & CA & \rightarrow AC & B & \rightarrow b \\
AB & \rightarrow BA & BC & \rightarrow CB & C & \rightarrow c \\
BA & \rightarrow AB & CB & \rightarrow BC
\end{align*}
\]

generated language: words with an equal number of \(a\), \(b\), and \(c\).
Example 2: type 0

Type 0:

\[ S \rightarrow SS' \$
\[ Aa \rightarrow aA \$
\[ Sa \rightarrow aS \$
\[ S' \rightarrow aAS' \$
\[ Ab \rightarrow bA \$
\[ S' \rightarrow bBS' \$
\[ Ba \rightarrow aB \$
\[ S' \rightarrow \varepsilon \$
\[ Bb \rightarrow bB \$
\[ S' \rightarrow \varepsilon \$
\[ Bb \rightarrow bB \$
\[ $\rightarrow $a
\[ $a \rightarrow a$ $b \rightarrow b$ $b \rightarrow b$
\[ $b \rightarrow b$
\[ $a \rightarrow a$
\[ $b \rightarrow b$
\[ $a \rightarrow a$
\[ $b \rightarrow b$
\[ $a \rightarrow a$
\[ $b \rightarrow b$
\[ $a \rightarrow a$
\[ $b \rightarrow b$
\[ $a \rightarrow a$
\[ $b \rightarrow b$
\[ $a \rightarrow a$
\[ $b \rightarrow b$
\[ $a \rightarrow a$
\[ $b \rightarrow b$
\[ $a \rightarrow a$
\[ $b \rightarrow b$
\[ $a \rightarrow a$
\[ $b \rightarrow b$
Example 2: type 0 (cont’d)

Formal Languages
Formal Grammars
Regular Languages
Formal complexity of Natural Languages
References

Definition
Language classes

Example 2: type 0 (cont’d)

Formal Languages
Formal Grammars
Regular Languages
Formal complexity of Natural Languages
References

Example 2: type 0 (cont’d)

Formal Languages
Formal Grammars
Regular Languages
Formal complexity of Natural Languages
References

Definition
Language classes

Example 2: type 0 (cont’d)

Formal Languages
Formal Grammars
Regular Languages
Formal complexity of Natural Languages
References

Definition
Language classes

Example 2: type 0 (cont’d)

Formal Languages
Formal Grammars
Regular Languages
Formal complexity of Natural Languages
References

Definition
Language classes

Example 2: type 0 (cont’d)
Language families

- **finite**
- **regular**
- **context-free**
- **context-sensitive**
- **Turing-decidable**
- **Turing-recognizable**
- **recursively enumerable**
- **recursive**
- **formal**

1. No constraint
2. Context-sensitive
3. Regular

Formal Languages  
Formal Grammars  
Regular Languages  
Formal complexity of Natural Languages  
References  

Definition  
Language classes
Remarks

- There are other ways to classify languages,
  - either on other properties of the grammars;
  - or on other properties of the languages
- Nested structures are preferred, but it’s not necessary
- When classes are nested, it is expected to have a growth of complexity/expressive power


Mannell, Robert. 1999. *Infinite number of sentences*. part of a set of class notes on the Internet.