Formal Languages applied to Linguistics

Pascal Amsili

Laboratoire Lattice, Université Sorbonne Nouvelle

Cogmaster, september 2019

General introduction

1. Mathematicians (incl. Chomsky) have formalized the notion of **language**

   It might be thought of as an oversimplification, but that's always the same story...

2. What does it buy us:
   1. Tools to think about theoretical issues about language(s) (expressiveness, complexity, comparability...)
   2. Tools to manipulate concretely language (e.g. with computers)
   3. A research programme:
      - Represent the syntax of natural language in a fully unambiguously specified way

Now let's get familiar with the mathematical notion of language

Overview

1. Formal Languages
   - Base notions
   - Definition
   - Problem

2. Formal Grammars

3. Regular Languages

4. Formal complexity of Natural Languages

Alphabet, word

Def. 1 (Alphabet)

An *alphabet* Σ 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 Σ is a finite sequence of letters from Σ. Formally, let $[p] = (1, 2, 3, 4, ..., p)$ (ordered integer sequence). Then a word is a *mapping*

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

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

Examples I

Alphabet $\{0,1,2,3,4,5,6,7,8,9,\cdot\}$

Words
$235 \cdot 29$
$007 \cdot 12$
$\cdot 1 \cdot 1 \cdot 00 \cdot \cdot$
~~$3 \cdot 1415962 \ldots$~~ (π)
...

Alphabet {▪, ▬}

Words
▬▬▬▪
▪
▪▪▬▪
...

Examples II

Alphabet {·– , –··· , –·–· , –·· , · , . . . }

Words ··· ––– ···

–·· ·· –·· · ·–· ––– –

·– –··· –··· ·– ·– –··· –··· ·–

. . .

Alphabet {a, woman, loves, man }

Words a

a woman loves a woman

man man a loves woman loves a

. . .

Monoid

Def. 3 (Σ^*)

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

It comprises a word with no letter, noted ε

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

N.B.: Σ^* is always infinite, except...

Monoid

Def. 3 (Σ^*)

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

It comprises a word with no letter, noted ε

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

N.B.: Σ^* is always infinite, except...
if $\Sigma = \emptyset$. Then $\Sigma^* = \{\varepsilon\}$.

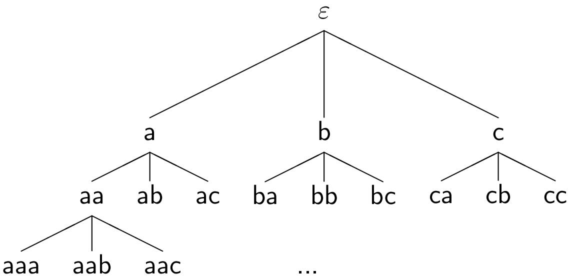
Structure of Σ^*

Let k be the size of the alphabet $k = |\Sigma|$.

Then Σ^* contains :	$k^0 = 1$	word(s) of 0 letters (ε)
	$k^1 = k$	word(s) of 1 letters
	k^2	word(s) of 2 letters
	...	
	k^n	words of n letters, $\forall n \geq 0$

Representation of Σ^*

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



- Words can be enumerated according to different orders
- Σ^* is a *countable* set

Concatenation

Σ^* 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:

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

Concatenation

Σ^* 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:

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

Example :	u	bacba
	v	cca

Concatenation

Σ^* 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:

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

Example :	u	bacba
	v	cca
	uv	bacbacca

Factor

Def. 5 (Factor)

A *factor* w of u is a subset of adjascent letters in u.

–w is a factor of u $\Leftrightarrow$ $\exists u_1, u_2$ s.t. $u = u_1 w u_2$

–w is a left factor (*prefix*) of u $\Leftrightarrow$ $\exists u_2$ s.t. $u = w u_2$

–w is a right factor (*suffix*) of u $\Leftrightarrow$ $\exists u_1$ 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 Σ.
   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:
   $a\,b\,a\,c\,c\,a\,b$

Role of concatenation

1. Words have been defined on Σ.
   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:
   $a\,b\,a\,c\,c\,a\,b$
   $(a\,b\,a)(c\,c\,a\,b)$

Role of concatenation

1. Words have been defined on Σ.
   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:
   $a\,b\,a\,c\,c\,a\,b$
   $(a\,b)(a\,c\,c)(a\,b)$

Role of concatenation

1. Words have been defined on Σ.
   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:
   $a\ b\ a\ c\ c\ a\ b$
   $(a\ b\ a\ c\ c)(a\ b)$

Role of concatenation

1. Words have been defined on Σ.
   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:
   $a\,b\,a\,c\,c\,a\,b$
   $(a)(b)(a)(c)(c)(a)(b)$

Role of concatenation

1. Words have been defined on Σ.
   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:
   $a\ b\ a\ c\ c\ a\ b$
   $(a)(b)(a)(c)(c)(a)(b)$
3. Since all letters of Σ form a word of length 1 (this set of words is called the *base*),
4. any word of Σ^* can be seen as a (unique) sequence of concatenations of length 1 words :
   $a\ b\ a\ c\ c\ a\ b$
   $((((((ab)a)c)c)a)b)$
   $((((((a.b).a).c).c).a).b)$

Properties of concatenation

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

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

Notation : $a.a.a = a^3$

Overview

1. Formal Languages
   - Base notions
   - Definition
   - Problem
2. Formal Grammars
3. Regular Languages
4. Formal complexity of Natural Languages

Language

Def. 7 ((Formal) Language)

Let Σ be an alphabet.
A language on Σ is a set of words on Σ.

Language

Def. 7 ((Formal) Language)

Let Σ be an alphabet.
A language on Σ is a set of words on Σ.
or, equivalently,
A language on Σ is a subset of Σ^*

Examples I

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

Examples I

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

Examples I

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

$L_1 = \{aa, ab, bac\}$	finite language
$L_2 = \{a, aa, aaa, aaaa \ldots\}$ or $L_2 = \{a^i \ / \ i \geq 1\}$	infinite language

Examples I

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

$L_1 = \{aa, ab, bac\}$	finite language
$L_2 = \{a, aa, aaa, aaaa \ldots\}$ or $L_2 = \{a^i \ / \ i \geq 1\}$	infinite language
$L_3 = \{\varepsilon\}$	finite language, reduced to a singleton

Examples I

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

$L_1 = \{aa, ab, bac\}$	finite language
$L_2 = \{a, aa, aaa, aaaa\ldots\}$ or $L_2 = \{a^i \ / \ i \geq 1\}$	infinite language
$L_3 = \{\varepsilon\}$	finite language, reduced to a singleton

$\neq$

Examples I

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

$L_1 = \{aa, ab, bac\}$	finite language
$L_2 = \{a, aa, aaa, aaaa \ldots\}$ or $L_2 = \{a^i \ / \ i \geq 1\}$	infinite language
$L_3 = \{\varepsilon\}$	finite language, reduced to a singleton
	$\neq$
$L_4 = \emptyset$	"empty" language

Examples I

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

$L_1 = \{aa, ab, bac\}$	finite language
$L_2 = \{a, aa, aaa, aaaa \ldots\}$ or $L_2 = \{a^i \ / \ i \geq 1\}$	infinite language
$L_3 = \{\varepsilon\}$	finite language, reduced to a singleton
	$\neq$
$L_4 = \emptyset$	"empty" language
$L_5 = \Sigma^*$	

Examples II

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

Examples II

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

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

$L = \{$ 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

$\Rightarrow$ One may describe a (complex) language as the result of set operations on (~~simpler~~) languages:
$\{a^{2k} \ / \ k \geqslant 1\} = \{a, aa, aaa, aaaa, \ldots\} \cap \{ww \ / \ w \in \Sigma^*\}$

Additional operations

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:

$$L_1.L_2 \quad = \quad \{uv \,/\, u \in L_1 \;\&\; v \in L_2\}$$

  Notation: $\overbrace{L.L.L\ldots L}^{k \text{ times}} = L^k \;;\; L^0 = \{\varepsilon\}$

- The Kleene star of a language is thus defined:

$$L^* = \bigcup_{n \geqslant 0} L^n$$

Regular expressions

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...

$(\{a\} \cup \{b\})^*.\{c\} = \{c, ac, abc, bc, \ldots, baabaac, \ldots\}$
(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...

$(\{a\} \cup \{b\})^*.\{c\} = \{c, ac, abc, bc, \ldots, baabaac, \ldots\}$
(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)
  - ⇒ Finite at a time t by hypothesis
- **words** well formed English sentences
  - ⇒ 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)

$\Rightarrow$ negligable

(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 profer a longuer 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

About infinity

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*

About infinity

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 !!

About infinity

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.

About infinity

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)

SORBONNE NOUVELLE

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 give algorithmic tools to recognize and to analyse words of a language.

recognize u	: decide whether $u \in L$
analyse u	: 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

Introduction

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

- Turing machines (automata)
- λ-terms
- ...

Formal grammar

Def. 9 ((Formal) Grammar)

A **formal grammar** is defined by $\langle \Sigma, N, S, P \rangle$ where

- Σ is an alphabet
- N is a disjoint alphabet non-terminal vocabulary)
- $S \in V$ is a distinguished elemnt 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

$$\langle \Sigma, N, S, P \rangle$$

$$\mathcal{G}_0 = \Bigg\langle$$

Examples

$$\langle \Sigma, N, S, P \rangle$$

$$\mathcal{G}_0 = \Bigg\langle \{joe, sam, sleeps\},$$

Examples

$$\langle \Sigma, N, S, P \rangle$$

$$\mathcal{G}_0 = \Bigg\langle \{joe, sam, sleeps\}, \{N, V, S\},$$

Examples

$$\langle \Sigma, N, S, P \rangle$$

$$\mathcal{G}_0 = \Bigg\langle \{joe, sam, sleeps\}, \{N, V, S\}, S,$$

Examples

$$\langle \Sigma, N, S, P \rangle$$

$$\mathcal{G}_0 = \left\langle \{joe, sam, sleeps\}, \{N, V, S\}, S, \left\{ \begin{array}{l} (N, joe) \\ (N, sam) \\ (V, sleeps) \\ (S, N\ V) \end{array} \right\} \right\rangle \}$$

Examples

$$\langle \Sigma, N, S, P \rangle$$

$$\mathcal{G}_0 = \left\langle \{joe, sam, sleeps\}, \{N, V, S\}, S, \left\{ \begin{array}{l} N \to joe \\ N \to sam \\ V \to sleeps \\ S \to N\ V \end{array} \right\} \right\rangle \}$$