Formal Languages and Linguistics

Pascal Amsili

Sorbonne Nouvelle, Lattice (CNRS/ENS-PSL/SN)

Cogmaster, september 2023

General introduction

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

   oversimplification ?
   maybe...

2. It buys us:
   2.1 Tools to think about theoretical issues about language/s (expressiveness, complexity, comparability...)
   2.2 Tools to manipulate concretely language (e.g. with computers)
   2.3 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

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 {·, —}

Words

— — — ·

·

· · — ·

...

Alphabet {·—, —···, —·—·, —··, ·, ...}

Words

··· ——— ···

—·· ·· —·· · ·—· ——— —

·— —··· —··· ·— ·— —··· —··· ·—

...

Examples II

Alphabet	$\{0,1,2,3,4,5,6,7,8,9, \cdot\}$
Words	$235 \cdot 29$
	$007 \cdot 12$
	$\cdot 1 \cdot 1 \cdot 00 \cdot \cdot$
	$\cancel{3 \cdot 1415962 \ldots}$ (π)
	...
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^*

Σ^* includes a word with no letter, noted ε

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

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^*

Σ^* includes a word with no letter, noted ε

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

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

Structure of Σ^*

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

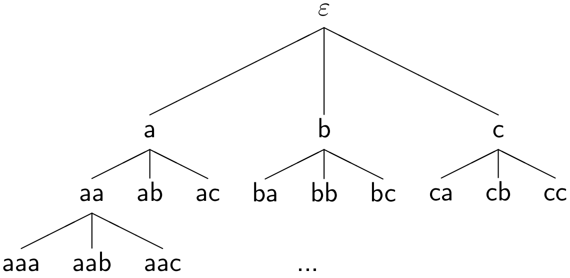
Then Σ^* contains :

$$\begin{array}{ll} k^0 = 1 & \text{word(s) of 0 letters } (\varepsilon) \\ k^1 = k & \text{word(s) of 1 letters} \\ k^2 & \text{word(s) of 2 letters} \\ \ldots & \\ k^n & \text{words of } n \text{ letters, } \forall n \geq 0 \end{array}$$

Sorbonne Nouvelle

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: *concatenation*

Def. 4 (Concatenation)

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

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

Sorbonne Nouvelle

Concatenation

Σ^* can be equipped with a binary operation: *concatenation*

Def. 4 (Concatenation)

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

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

Example :	u	bacba
	v	cca

Concatenation

Σ^* can be equipped with a binary operation: *concatenation*

Def. 4 (Concatenation)

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

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

Example :	u	bacba
	v	cca
	uv	bacbacca

Sorbonne Nouvelle

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 into factors.

Role of concatenation

1. Words have been defined on Σ.
   Given any two 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.
   Given any two 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.
   Given any two 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.
   Given any two 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.
   Given any two 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.
   Given any two 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

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

Sorbonne Nouvelle

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

Sorbonne Nouvelle

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

Examples II

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

Examples II

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

$L = \{$ 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:
$\{a^{2k} \ / \ k \geqslant 1\} = \{a, aa, aaa, aaaa, \ldots\} \cap \{ww \ / \ w \in \Sigma^*\}$

Sorbonne Nouvelle

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

Overview

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

Good questions

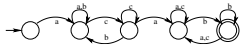
Why would one consider natural language as a formal language?

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

recognize u	: decide whether $u \in L$
analyse u	: show the internal structure of u

Sorbonne Nouvelle

Final remarks

- We are only talking about syntax
- From now on, we'll mostly be looking for precise and efficient ways to **define** a language
  - $\mathrm{L} = \{aa, ab, ba\}$
  - $\mathrm{L} = \{$ all the country names in English $\}$
  - $\mathrm{L} = \{$ all the inflected forms of French *manger* $\}$
  - $\mathrm{L} = \{a^{2^k} \text{ with } k \geq 0\}$
  - $\mathrm{L} = \{ww \text{ with } w \in \Sigma^*\}$
  - $\mathrm{L} = (\{a\} \cup \{b\}.\{c\})^*$ — simplified notation $(a|bc)^*$
  - L = the set of words recognized by this automaton: 
  - L = the set of words engendered by this formal grammar

$$
\begin{array}{llll}
\mathcal{G}_3: & E & \longrightarrow & E + E \\
 & & | & E \times E \\
 & & | & (\, E \,) \\
 & & | & F \\
 & F & \longrightarrow & 0|1|2|3|4|5|6|7|8|9
\end{array}
$$

Sorbonne Nouvelle

Overview

Definition

3 possible definitions

1. a regular language can be defined by rational/regular expressions
2. a regular language can be recognized by a finite automaton
3. a regular language can be generated by a regular grammar

Overview

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)

Sorbonne Nouvelle

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.

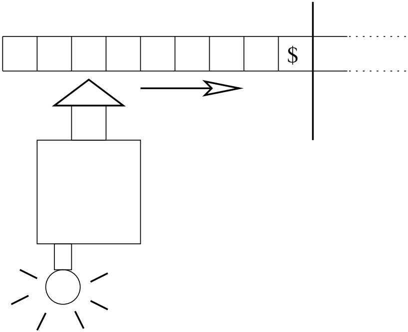
Def. 9 (Rational Language)

A rational language on Σ is a subset of Σ^* inductively defined thus:

- $\emptyset$ and $\{\varepsilon\}$ are rational languages ;
- for all $a \in X$, the singleton $\{a\}$ is a rational language ;
- for all g and h rational, the sets $g \cup h$, $g.h$ and g^* are rational languages.

Overview

Metaphoric definition



Formal definition

Def. 10 (Finite deterministic automaton (FDA))

A finite state deterministic automaton $\mathcal{A}$ is defined by :

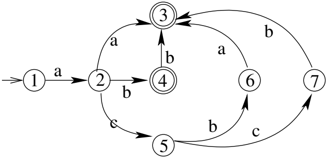
$$\mathcal{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$$

- Q is a finite set of states
- Σ is an alphabet
- q_0 is a distinguished state, the initial state,
- F is a subset of Q, whose members are called final/terminal states
- δ is a mapping **fonction** from $Q \times \Sigma$ to Q. Notation $\delta(q, a) = r$.

Example

Let us consider the (finite) language $\{aa, ab, abb, acba, accb\}$.
The following automaton recognizes this langage: $\langle Q, \Sigma, q_0, F, \delta \rangle$, avec $Q = \{1, 2, 3, 4, 5, 6, 7\}$, $\Sigma = \{a, b, c\}$, $q_0 = 1$, $F = \{3, 4\}$, and δ is thus defined:

$$\delta : \begin{array}{l} (1,\text{a}) \mapsto 2 \\ (2,\text{a}) \mapsto 3 \\ (2,\text{b}) \mapsto 4 \\ (2,\text{c}) \mapsto 5 \\ (4,\text{b}) \mapsto 3 \\ (5,\text{b}) \mapsto 6 \\ (5,\text{c}) \mapsto 7 \\ (6,\text{a}) \mapsto 3 \\ (7,\text{b}) \mapsto 3 \end{array}$$



	a	b	c
$\rightarrow 1$	2		
2	3	4	5
$\leftarrow 3$			
$\leftarrow 4$		3	
5		6	7
6	3		
7		3	

Recognition

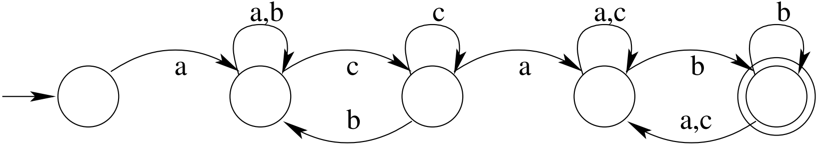
Recognition is defined as the existence of a sequence of states defined in the following way. Such a sequence is called a path in the automaton.

Def. 11 (Recognition)

A word $a_1 a_2 ... a_n$ is **recognized/accepted** by an automaton iff there exists a sequence $k_0, k_1, ..., k_n$ of states such that:

$$k_0 = q_0$$
$$k_n \in F$$
$$\forall i \in [1, n], \;\; \delta(k_{i-1}, a_i) = k_i$$

Example



Exercices

Let $\Sigma = \{a, b, c\}$. Give deterministic finite state automata that accept the following languages:

1. The set of words with an even length.
2. The set of words where the number of occurrences of b is divisible by 3.
3. The set of words ending with a b.
4. The set of words not ending with a b.
5. The set of words non empty not ending with a b.
6. The set of words comprising at least a b.
7. The set of words comprising at most a b.
8. The set of words comprising exactly one b.

References I

Bar-Hillel, Yehoshua, Perles, Micha, & Shamir, Eliahu. 1961. On formal properties of simple phrase structure grammars. *STUF-Language Typology and Universals*, **14**(1-4), 143–172.

Chomsky, Noam. 1957. *Syntactic Structures*. Den Haag: Mouton & Co.

Chomsky, Noam. 1995. *The Minimalist Program*. Vol. 28. Cambridge, Mass.: MIT Press.

Gazdar, Gerald, & Pullum, Geoffrey K. 1985 (May). *Computationally Relevant Properties of Natural Languages and Their Grammars*. Tech. rept. Center for the Study of Language and Information, Leland Stanford Junior University.

Gibson, Edward, & Thomas, James. 1997. The Complexity of Nested Structures in English: Evidence for the Syntactic Prediction Locality Theory of Linguistic Complexity. *Unpublished manuscript, Massachusetts Institute of Technology*.

Joshi, Aravind K. 1985. *Tree Adjoining Grammars: How Much Context-Sensitivity is Required to Provide Reasonable Structural Descriptions?* Tech. rept. Department of Computer and Information Science, University of Pennsylvania.

Langendoen, D Terence, & Postal, Paul Martin. 1984. *The vastness of natural languages*. Basil Blackwell Oxford.

Mannell, Robert. 1999. *Infinite number of sentences*. part of a set of class notes on the Internet. `http://clas.mq.edu.au/speech/infinite_sentences/`.

Schieber, Stuart M. 1985. Evidence against the Context-Freeness of Natural Language. *Linguistics and Philosophy*, **8**(3), 333–343.

Stabler, Edward P. 2011. Computational perspectives on minimalism. *Oxford handbook of linguistic minimalism*, 617–643.

Steedman, Mark, *et al.* . 2012 (June). *Combinatory Categorial Grammars for Robust Natural Language Processing*. Slides for NASSLLI course `http://homepages.inf.ed.ac.uk/steedman/papers/ccg/nasslli12.pdf`.

Vijay-Shanker, K., & Weir, David J. 1994. The Equivalence of Four Extensions of Context–Free Grammars. *Mathematical Systems Theory*, **27**, 511–546.