Formal Languages and Linguistics

Pascal Amsili

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

Cogmaster, september 2022

Overview

Formal Languages

Regular Languages

- Definition
- Regular expressions
- Automata
- Properties

Formal Grammars

Formal complexity of Natural Languages

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.

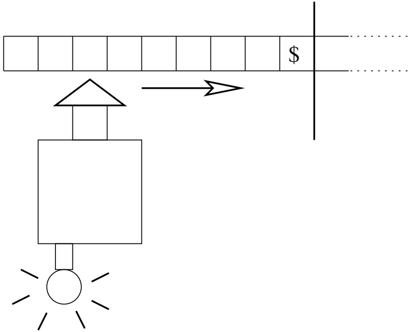
Def. 9 (Rational Language)

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 :

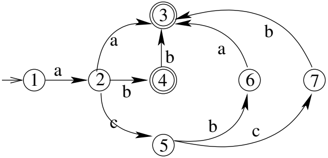
$$\mathcal{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$$

- 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:

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



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

Recognition

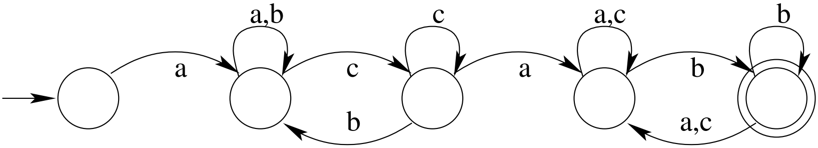
Recognition is defined as the existence of a sequence of states defined in the following way. Such a sequence is called a path in the automaton.

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:

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

Example

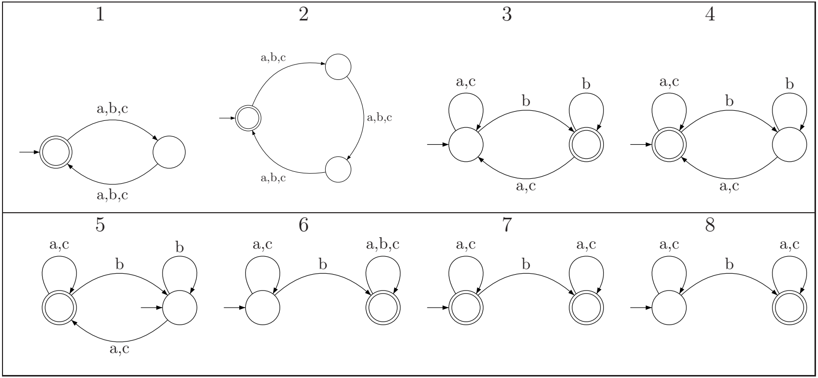


Exercices

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.

Answers



Overview

Ways of non-determinism

A word is recognized if there exists a path in the automaton. It is not excluded however that there be several paths for one word: in that case, the automaton is non deterministic.
What are the sources of non determinism?

- $\delta(a, S_1) = \{S_2, S_3\}$
- "spontaneous transition" = ε-transition

Equivalence theorems

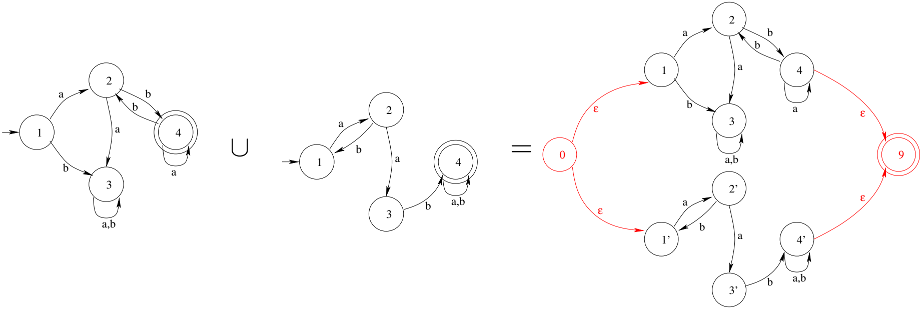
For any non-deterministic automaton, it is possible to design a complete deterministic automaton that recognizes the same language.
Proofs: algorithms (constructive proofs)
First “remove” ε-transitions, then “remove” multiple transitions.

Closure (1)

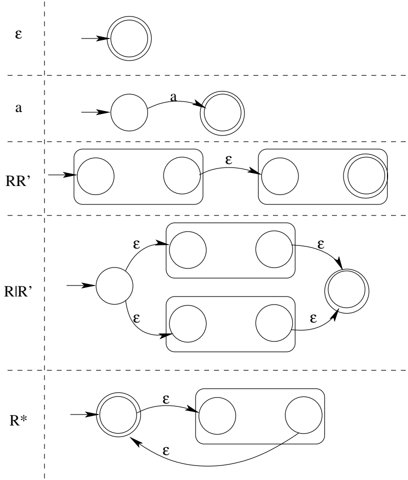
Regular languages are closed under various operations: if the languages L and L' are regular, so are:

- $L \cup L'$ (union); $L.L'$ (product); L^* (Kleene star)
(rational operations)

Union of regular languages: an example



Rational operations



Closure (2)

Regular languages are closed under various operations: if the languages L and L' are regular, so are:

- $L \cup L'$ (union); $L.L'$ (product); L^* (Kleene star)
  (rational operations)
  $\rightarrow$ for every rational expression describing a language , there is a FSA that recognizes L

Closure (2)

Regular languages are closed under various operations: if the languages L and L' are regular, so are:

- $L \cup L'$ (union); $L.L'$ (product); L^* (Kleene star)
  (rational operations)
  $\rightarrow$ for every rational expression describing a language , there is a FSA that recognizes L and vice-versa

Closure (2)

Regular languages are closed under various operations: if the languages L and L' are regular, so are:

- $L \cup L'$ (union); $L.L'$ (product); L^* (Kleene star) *(rational operations)*
  $\rightarrow$ for every rational expression describing a language , there is a FSA that recognizes L and vice-versa
- $L \cap L'$ (intersection); $\overline{L}$ (complement)
- ...

Intersection of regular languages

Algorithmic proof
Deterministic complete automata

L_1	a	b
$\rightarrow$ 1	2	4
2	4	3
$\leftarrow$ 3	3	3
4	4	4

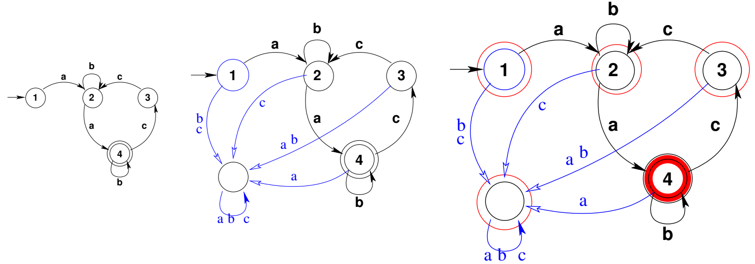
L_2	a	b
$\leftrightarrow$ 1	2	5
2	5	3
3	4	5
4	1	4
5	5	5

$L_1 \cap L_2$	a	b
$\rightarrow$ (1,1)	(2,2)	(4,5)
(2,2)	(4,5)	(3,3)
(4,5)	(4,5)	(4,5)
(3,3)	(3,4)	(3,5)
(3,4)	(3,1)	(3,4)
$\leftarrow$ (3,1)	(3,2)	(3,4)
(3,2)	(3,4)	(3,3)
(3,5)	(3,5)	(3,5)

Complement of a regular language

Deterministic complete automata

completed complemented



Pumping lemma: Intuition

Take an automaton with k states.

Pumping lemma: Intuition

Take an automaton with k states.
If the accepted language is infinite,
then some words have more than k letters.

Pumping lemma: Intuition

Take an automaton with k states.
If the accepted language is infinite,
then some words have more than k letters.
Therefore, at least one state has to be "gone through" several times.

Pumping lemma: Intuition

Take an automaton with k states.
If the accepted language is infinite,
then some words have more than k letters.
Therefore, at least one state has to be "gone through" several times.
That means there is a loop on that state.

Pumping lemma: Intuition

Take an automaton with k states.
If the accepted language is infinite,
then some words have more than k letters.
Therefore, at least one state has to be "gone through" several times.
That means there is a loop on that state.
Then making any number of loops will end up with a word in L.

$\Rightarrow$ Pumping lemma

Pumping lemma: definition

Def. 12 (Pumping Lemma)

Let L be an infinite regular language.
There exists an integer k such that:

$$\begin{array}{rl} \multicolumn{2}{l}{\forall x \in L,\ |x| > k,\ \exists u, v, w \ \text{ such that } x = uvw,\ \text{with:}} \\ (i) & |v| \geq 1 \\ (ii) & |uv| \leq k \\ (iii) & \forall i \geq 0,\ uv^i w \in L \end{array}$$

Pumping lemma: Illustration

Let's illustrate the lemma with a language which trivialy satisfies it: a^*bc.
Let $k = 3$, the work abc is long enough, and can be decomposed:

$$\underbrace{\varepsilon}_{u}\ \underbrace{a}_{v}\ \underbrace{b\ c}_{w}$$

The three properties of the lemma are satisfied:

- $|v| \geq 1$ $(v = a)$
- $|uv| \leq k$ $(uv = a)$
- $\forall i \in \mathbb{N}$, $uv^i w (= a^i bc)$ belongs to the language by definition.

Pumping lemma: Consequences

The pumping lemma is a tool to prove that a language is **not** regular.

$\mathcal{L}$ regular	$\Rightarrow$	pumping lemma $(\forall i, uv^i w \in \mathcal{L})$
pumping lemma	$\not\Rightarrow$	$\mathcal{L}$ regular
NO pumping lemma	$\Rightarrow$	$\mathcal{L}$ **NOT** regular

Pumping lemma: Consequences

The pumping lemma is a tool to prove that a language is **not** regular.

$\mathcal{L}$ regular	$\Rightarrow$	pumping lemma $(\forall i, uv^iw \in \mathcal{L})$
pumping lemma	$\not\Rightarrow$	$\mathcal{L}$ regular
NO pumping lemma	$\Rightarrow$	$\mathcal{L}$ **NOT** regular

to prove that $\mathcal{L}$ is

- regular: provide an automaton
- not regular: show that the pumping lemma does not apply

Pumping lemma: Consequences

Def. 13 (Consequences)

Let $\mathcal{A}$ be a k state automaton:

1. $L(\mathcal{A}) \neq \emptyset$ ***iff*** $\mathcal{A}$ recognises (at least) one word u s.t. $|u| < k$.
2. $L(\mathcal{A})$ is infinite ***iff*** $\mathcal{A}$ recognises (at least) one word u t.q. $k \leq |u| < 2k$.

Results: expressivity

- Any finite langage is regular
- $a^n b^m$ is regular
- $a^n b^n$ is not regular
- ww^R is not regular (^{R} : reverse word)

Decidable problems

- The "word problem" $w \overset{?}{\in} L(\mathcal{A})$ is decidable.

$\Rightarrow$ A computation on an automaton always stops.

Decidable problems

- The "word problem" $w \overset{?}{\in} L(\mathcal{A})$ is decidable.

$\Rightarrow$ A computation on an automaton always stops.

- The "emptiness problem" $L(\mathcal{A}) \overset{?}{=} \emptyset$ is decidable.

$\Rightarrow$ It's enough to test all possible words of length $\leq k$, where k is the number of states.

Decidable problems

- The "word problem" $w \stackrel{?}{\in} L(\mathcal{A})$ is decidable.

$\Rightarrow$ A computation on an automaton always stops.

- The "emptiness problem" $L(\mathcal{A}) \stackrel{?}{=} \emptyset$ is decidable.

$\Rightarrow$ It's enough to test all possible words of length $\leq k$, where k is the number of states.

- The "finiteness problem" $L(\mathcal{A}) \stackrel{?}{\text{is}}$ *finite* is decidable.

$\Rightarrow$ Test all possible words whose length is between k and $2k$. If there exists u s.t. $k < |u| < 2k$ and $u \in L(\mathcal{A})$, then $L(\mathcal{A})$ is infinite.

Decidable problems

- The "word problem" $w \overset{?}{\in} L(\mathcal{A})$ is decidable.

$\Rightarrow$ A computation on an automaton always stops.

- The "emptiness problem" $L(\mathcal{A}) \overset{?}{=} \emptyset$ is decidable.

$\Rightarrow$ It's enough to test all possible words of length $\leq k$, where k is the number of states.

- The "finiteness problem" $L(\mathcal{A})$ $\overset{?}{\text{is}}$ *finite* is decidable.

$\Rightarrow$ Test all possible words whose length is between k and $2k$. If there exists u s.t. $k < |u| < 2k$ and $u \in L(\mathcal{A})$, then $L(\mathcal{A})$ is infinite.

- The "equivalence problem" $L(\mathcal{A}) \overset{?}{=} L(\mathcal{A}')$ is decidable.

$\Rightarrow$ it boils down to answering the question:

$$\left(L(\mathcal{A}) \cap \overline{L(\mathcal{A}')}\right) \cup \left(L(\mathcal{A}') \cap \overline{L(\mathcal{A})}\right) = \emptyset$$

À quoi ça sert?

Why would you want to define (formally) a language?

- to formulate a request to a search engine (`mang.*`)
- to associate actions to (classes of) words (e.g., transducers)
  - formal languages (math. expressions, programming languages...)
  - artificial (interface) languages
  - (subpart of) natural languages

Overview

Formal grammar

Def. 14 ((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 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

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

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

Examples

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

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

Examples

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

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

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

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

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

Examples (cont'd)

$$\mathcal{G}_1 = \left\langle \{jean, dort\}, \{Np, SN, SV, V, S\}, S, \left\{ \begin{array}{l} S \rightarrow SN\ SV \\ SN \rightarrow Np \\ SV \rightarrow V \\ Np \rightarrow jean \\ V \rightarrow dort \end{array} \right\} \right\rangle \}$$

$$\mathcal{G}_2 = \langle \{(,)\}, \{S\}, S, \{S \longrightarrow \varepsilon \,|\, (S)S\} \rangle$$

Notation

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

Notation

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

$$\mathcal{G}_3 = \langle \{+, \times, (,), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \{E, F\}, E, \{\ldots\} \rangle$$

Notation

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

$$\mathcal{G}_3 = \langle \{+, \times, (,), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \{E, F\}, E, \{\ldots\} \rangle$$

$$G_4 = E \rightarrow E + T \mid T, T \rightarrow T \times F \mid F, F \rightarrow (E) \mid a$$

Immediate Derivation

Def. 15 (Immediate derivation)

Let $\mathcal{G} = \langle X, V, S, P \rangle$ a grammar, $(f, g) \in (X \cup V)^*$ two "words", $r \in P$ a production rule, such that $r : A \longrightarrow u$ $(u \in (X \cup V)^*)$.

- f derives into g (immediate derivation) with the rule r (noted $f \xrightarrow{r} g$) iff
  $\exists v, w$ s.t. $f = vAw$ and $g = vuw$
- f derives into g (immediate derivation) in the grammar $\mathcal{G}$ (noted $f \xrightarrow{\mathcal{G}} g$) iff
  $\exists r \in P$ s.t. $f \xrightarrow{r} g$.

Derivation

Def. 16 (Derivation)

$$f \xrightarrow{\mathcal{G}*} g \text{ if } \begin{array}{ll} f = g & \text{or} \\ \exists f_0, f_1, f_2, \ldots, f_n \text{ s.t. } & f_0 = f \\ & f_n = g \\ & \forall i \in [1, n] : f_{i-1} \xrightarrow{\mathcal{G}} f_i \end{array}$$

An example with $\mathcal{G}_0$:
N V joe N

Derivation

Def. 16 (Derivation)

$f \xrightarrow{\mathcal{G}*} g$ if $f = g$ or

$\exists f_0, f_1, f_2, ..., f_n$ s.t. $f_0 = f$

$f_n = g$

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

An example with $\mathcal{G}_0$:

$N\ V\ joe\ N \longrightarrow sam\ V\ joe\ N$

Derivation

Def. 16 (Derivation)

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

An example with $\mathcal{G}_0$:

$N\ V\ joe\ N \longrightarrow sam\ V\ joe\ N \longrightarrow\ sam\ V\ joe\ joe$ or

Derivation

Def. 16 (Derivation)

$$f \xrightarrow{\mathcal{G}*} g \text{ if } \begin{cases} f = g & \text{or} \\ \exists f_0, f_1, f_2, \dots, f_n \text{ s.t. } \begin{array}{l} f_0 = f \\ f_n = g \\ \forall i \in [1, n] : f_{i-1} \xrightarrow{\mathcal{G}} f_i \end{array} \end{cases}$$

An example with $\mathcal{G}_0$:

$$N\ V\ joe\ N \longrightarrow sam\ V\ joe\ N \longrightarrow \begin{array}{ll} sam\ V\ joe\ joe & \text{or} \\ sam\ V\ joe\ sam & \text{or} \end{array}$$

Derivation

Def. 16 (Derivation)

$$f \xrightarrow{\mathcal{G}*} g \text{ if } \begin{cases} f = g & \text{or} \\ \exists f_0, f_1, f_2, ..., f_n \text{ s.t. } & f_0 = f \\ & f_n = g \\ & \forall i \in [1, n] : f_{i-1} \xrightarrow{\mathcal{G}} f_i \end{cases}$$

An example with $\mathcal{G}_0$:

$$N\ V\ joe\ N \longrightarrow sam\ V\ joe\ N \longrightarrow \begin{array}{ll} sam\ V\ joe\ joe & \text{or} \\ sam\ V\ joe\ sam & \text{or} \\ sam\ sleeps\ joe\ N & \text{or} \\ \ldots & \end{array}$$

Endpoint of a derivation

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

An example with $\mathcal{G}_3$:

$E \times E$

Endpoint of a derivation

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

An example with $\mathcal{G}_3$:

$$E \times E \longrightarrow F \times E$$

Endpoint of a derivation

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

An example with $\mathcal{G}_3$:

$$E \times E \longrightarrow F \times E \longrightarrow 3 \times E$$

Endpoint of a derivation

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

An example with $\mathcal{G}_3$:

$$E \times E \longrightarrow F \times E \longrightarrow 3 \times E \longrightarrow 3 \times (E)$$

Endpoint of a derivation

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

An example with $\mathcal{G}_3$:

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

Endpoint of a derivation

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

An example with $\mathcal{G}_3$:

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

Endpoint of a derivation

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

An example with $\mathcal{G}_3$:

$$E \times E \longrightarrow F \times E \longrightarrow 3 \times E \longrightarrow 3 \times (E) \longrightarrow 3 \times (E + E) \longrightarrow 3 \times (E + F) \longrightarrow 3 \times (E + 4)$$

Endpoint of a derivation

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

An example with $\mathcal{G}_3$:

$$E \times E \longrightarrow F \times E \longrightarrow 3 \times E \longrightarrow 3 \times (E) \longrightarrow 3 \times (E+E) \longrightarrow$$
$$3 \times (E+F) \longrightarrow 3 \times (E+4) \longrightarrow 3 \times (F+4)$$

Endpoint of a derivation

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

An example with $\mathcal{G}_3$:

$$
E \times E \longrightarrow F \times E \longrightarrow 3 \times E \longrightarrow 3 \times (E) \longrightarrow 3 \times (E + E) \longrightarrow
3 \times (E + F) \longrightarrow 3 \times (E + 4) \longrightarrow 3 \times (F + 4) \longrightarrow 3 \times (5 + 4)
$$

Endpoint of a derivation

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

An example with $\mathcal{G}_3$:

$$
E \times E \longrightarrow F \times E \longrightarrow 3 \times E \longrightarrow 3 \times (E) \longrightarrow 3 \times (E+E) \longrightarrow \\
3 \times (E+F) \longrightarrow 3 \times (E+4) \longrightarrow 3 \times (F+4) \longrightarrow 3 \times (5+4) \longrightarrow\!\!|
$$

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$:

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$: $S \rightarrow (S)S$

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$: $S \to (S)S \to ()S$

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$: $S \rightarrow (S)S \rightarrow ()S \rightarrow ()$

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$: $S \rightarrow (S)S \rightarrow ()S \rightarrow ()$
as well as $((()))$, $()()()$, $((()(())))\ldots$

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$: $S \rightarrow (S)S \rightarrow ()S \rightarrow ()$
as well as $((()))$, $()()()$, $((()()()))\dots$
but $)()(\notin L_{\mathcal{G}_2}$, even though the following is a licit derivation :

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$: $S \rightarrow (S)S \rightarrow ()S \rightarrow ()$
as well as $((()))$, $()()()$, $((()()()))\ldots$
but $)()(\notin L_{\mathcal{G}_2}$, even though the following is a licit derivation :
$)S(\rightarrow$

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$: $S \rightarrow (S)S \rightarrow ()S \rightarrow ()$
as well as $((()))$, $()()()$, $((()()()))\ldots$
but $)()(\notin L_{\mathcal{G}_2}$, even though the following is a licit derivation :
$)S(\rightarrow)(S)S(\rightarrow$

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$: $S \rightarrow (S)S \rightarrow ()S \rightarrow ()$
as well as $((()))$, $()()()$, $((()()()))\ldots$
but $)()(\notin L_{\mathcal{G}_2}$, even though the following is a licit derivation :
$)S(\rightarrow)(S)S(\rightarrow)()S(\rightarrow$

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$: $S \rightarrow (S)S \rightarrow ()S \rightarrow ()$
as well as $((()))$, $()()()$, $((()()()))\ldots$
but $)()(\notin L_{\mathcal{G}_2}$, even though the following is a licit derivation :
$)S(\rightarrow)(S)S(\rightarrow)()S(\rightarrow)()($

Engendered language

Def. 17 (Language engendered by a word)

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

$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 18 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

For instance $() \in L_{\mathcal{G}_2}$: $S \rightarrow (S)S \rightarrow ()S \rightarrow ()$
as well as $((())), ()()(), ((()()()))\ldots$
but $)()(\notin L_{\mathcal{G}_2}$, 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

$$G_4 = E \rightarrow E + T \mid T, T \rightarrow T \times F \mid F, F \rightarrow (E) \mid a$$

$a + a$, $a + (a \times a)$, ...

Proto-word

Def. 19 (Proto-word)

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

$$E \rightarrow E + T \rightarrow E + T * F \rightarrow T + T * F \rightarrow T + F * F \rightarrow T + a * F \rightarrow F + a * F \rightarrow a + a * F \rightarrow \cancel{a + a * a}$$

Multiple derivations

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

Multiple derivations

A given word may have several derivations:

$E \rightarrow E + E \rightarrow F + E \rightarrow F + F \rightarrow 3 + F \rightarrow 3 + 4$

$E \rightarrow E + E \rightarrow E + F \rightarrow E + 4 \rightarrow F + 4 \rightarrow 3 + 4$

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:

Multiple derivations

A given word may have several derivations:
$E \to E + E \to F + E \to F + F \to 3 + F \to 3 + 4$
$E \to E + E \to E + F \to E + 4 \to F + 4 \to 3 + 4$
... but if the grammar is not ambiguous, there is only one **left** derivation:
$\underline{E} \to \underline{E} + E \to \underline{F} + E \to 3 + \underline{E} \to 3 + \underline{F} \to 3 + 4$

Multiple derivations

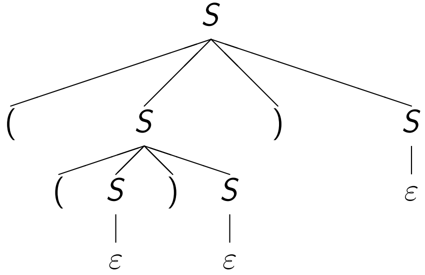
A given word may have several derivations:
$E \rightarrow E+E \rightarrow F+E \rightarrow F+F \rightarrow 3+F \rightarrow 3+4$
$E \rightarrow E+E \rightarrow E+F \rightarrow E+4 \rightarrow F+4 \rightarrow 3+4$
... but if the grammar is not ambiguous, there is only one **left** derivation:
$\underline{E} \rightarrow \underline{E}+E \rightarrow \underline{F}+E \rightarrow 3+\underline{E} \rightarrow 3+\underline{F} \rightarrow 3+4$

parsing: trying to find the/a left derivation (resp. right)

Derivation tree

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.

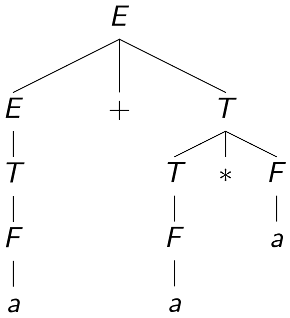
Grammar $\mathcal{G}_2$:
$$\begin{array}{rcl} S & \longrightarrow & \varepsilon \\ & | & (S)S \end{array}$$



$$S \rightarrow (S)S \rightarrow ((S)S)S \rightarrow ((S)S) \rightarrow ((S)) \rightarrow (())$$

Structural analysis

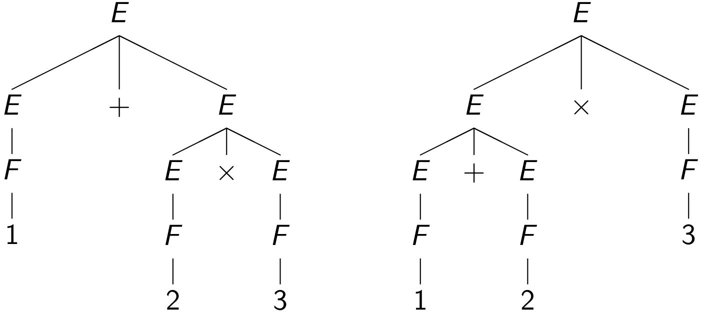
Syntactic trees are precious to give access to the semantics



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, $\mathcal{G}_3$ is ambiguous, since it can assign the two follwing trees to $1 + 2 \times 3$:



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) $\{a^n ba^m ba^p ba^q | (n \geqslant q \wedge m \geqslant p) \vee (n \geqslant m \wedge p \geqslant q)\}$

Comparison of grammars

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

Overview

Formal Languages

Regular Languages

Formal Grammars

Definition

Language classes

Formal complexity of Natural Languages

Principle

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 $P \subset (X \cup V)^* V (X \cup V)^* \times (X \cup V)^*$.

type 1 (*context-sensitive* grammars) All rules of P are of the shape $(u_1 S u_2, u_1 m u_2)$, 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:

$$
\begin{array}{lcl}
S & \rightarrow & aS \mid aB \mid bB \mid cA \\
B & \rightarrow & bB \mid b \\
A & \rightarrow & cS \mid bB
\end{array}
$$

Examples

type 3:

$$\begin{array}{lcl} S & \rightarrow & aS \mid aB \mid bB \mid cA \\ B & \rightarrow & bB \mid b \\ A & \rightarrow & cS \mid bB \end{array}$$

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:

$$\begin{array}{llll} S & \rightarrow SABC & AC \rightarrow CA & A \rightarrow a \\ S & \rightarrow \varepsilon & CA \rightarrow AC & B \rightarrow b \\ AB & \rightarrow BA & BC \rightarrow CB & C \rightarrow c \\ BA & \rightarrow AB & CB \rightarrow BC & \end{array}$$

generated language :

Example 1 type 0

Type 0:

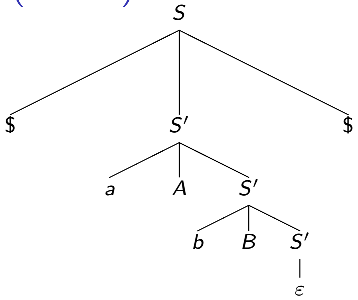
$$
\begin{array}{llll}
S & \rightarrow SABC & AC \rightarrow CA & A \rightarrow a \\
S & \rightarrow \varepsilon & CA \rightarrow AC & B \rightarrow b \\
AB & \rightarrow BA & BC \rightarrow CB & C \rightarrow c \\
BA & \rightarrow AB & CB \rightarrow BC &
\end{array}
$$

generated language : words with an equal number of a, b, and c.

Example 2: type 0

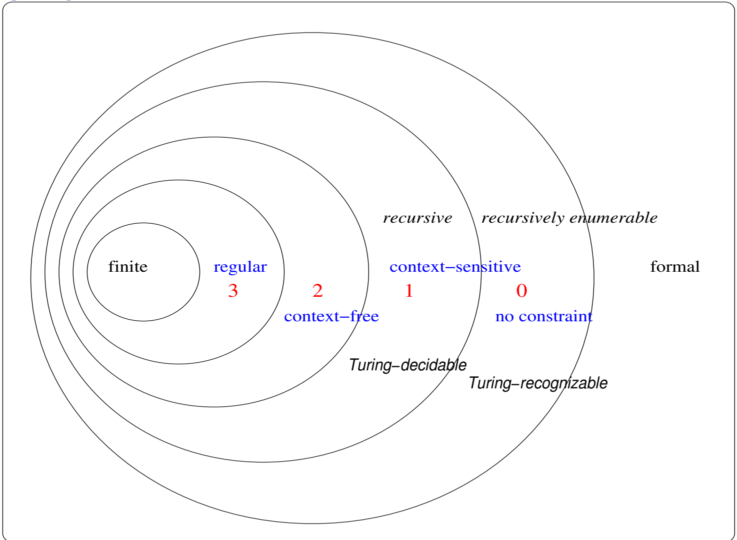
$$
\text{Type 0:}\quad
\begin{array}{lll}
S \rightarrow \$S'\$ & Aa \rightarrow aA & \$a \rightarrow a\$ \\
S' \rightarrow aAS' & Ab \rightarrow bA & \$b \rightarrow b\$ \\
S' \rightarrow bBS' & Ba \rightarrow aB & A\$ \rightarrow \$a \\
S' \rightarrow \varepsilon & Bb \rightarrow bB & B\$ \rightarrow \$b \\
 & & \$\$ \rightarrow \#
\end{array}
$$

Example 2: type 0 (cont'd)



\$	a	A	b	B	\$
a	\$	A	b	B	\$
a	\$	A	b	\$	b
a	\$	b	A	\$	b
a	b	\$	A	\$	b
a	b	\$	\$	a	b
a	b	#		a	b

Language families



Remarks

- There are others 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