Formal Languages and Linguistics

Pascal Amsili

Sorbonne Nouvelle, Lattice (CNRS/ENS-PSL)

Cogmaster, september 2020

Sorbonne Nouvelle

Overview

1. Formal Languages
2. Formal Grammars
   - Definition
   - Language classes
3. Regular Languages
4. 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:

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

Examples

type 3:

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

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:

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

generated language :

Example 1 type 0

Type 0:

$S \to SABC$	$AC \to CA$	$A \to a$
$S \to \varepsilon$	$CA \to AC$	$B \to b$
$AB \to BA$	$BC \to CB$	$C \to c$
$BA \to AB$	$CB \to BC$	

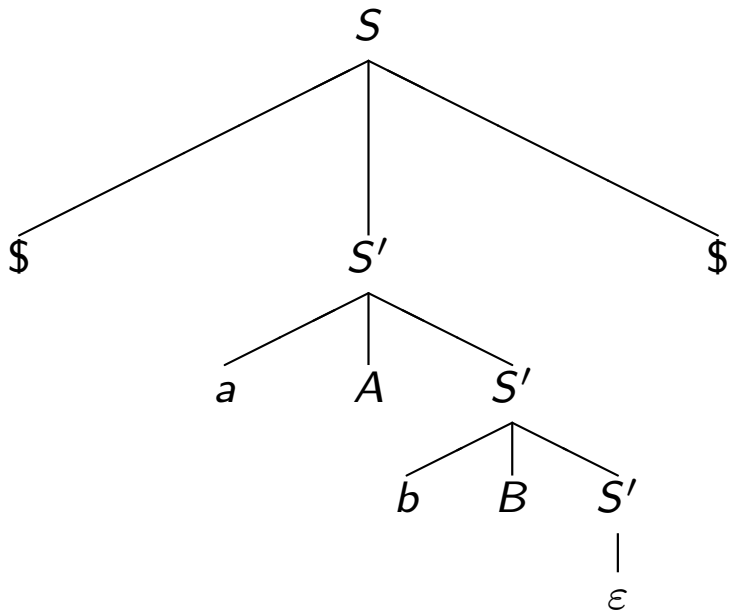
generated language : words with an equal number of a, b, and c.

Example 2: type 0

Type 0:

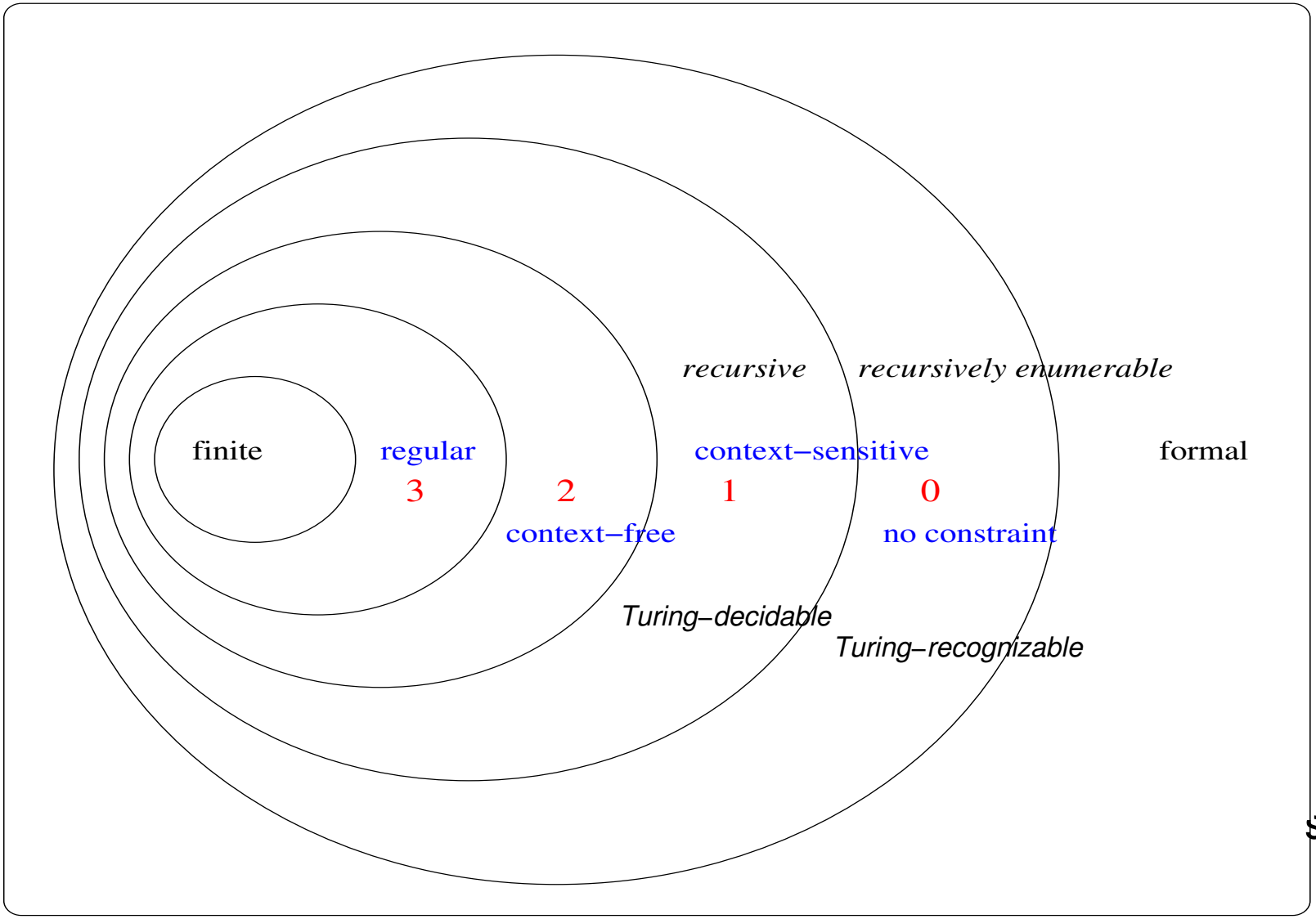
$$
\begin{array}{llll}
S \to \$S'\$ & Aa \to aA & \$a \to a\$ \\
S' \to aAS' & Ab \to bA & \$b \to b\$ \\
S' \to bBS' & Ba \to aB & A\$ \to \$a \\
S' \to \varepsilon & Bb \to bB & B\$ \to \$b \\
 & & \$\$ \to \#
\end{array}
$$

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

Overview

1. Formal Languages
2. Formal Grammars
3. Regular Languages
   - Definition
   - Automata
   - Properties
4. Formal complexity of Natural Languages

Definition

3 possible definitions

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

Def. 15 (Rational Language)

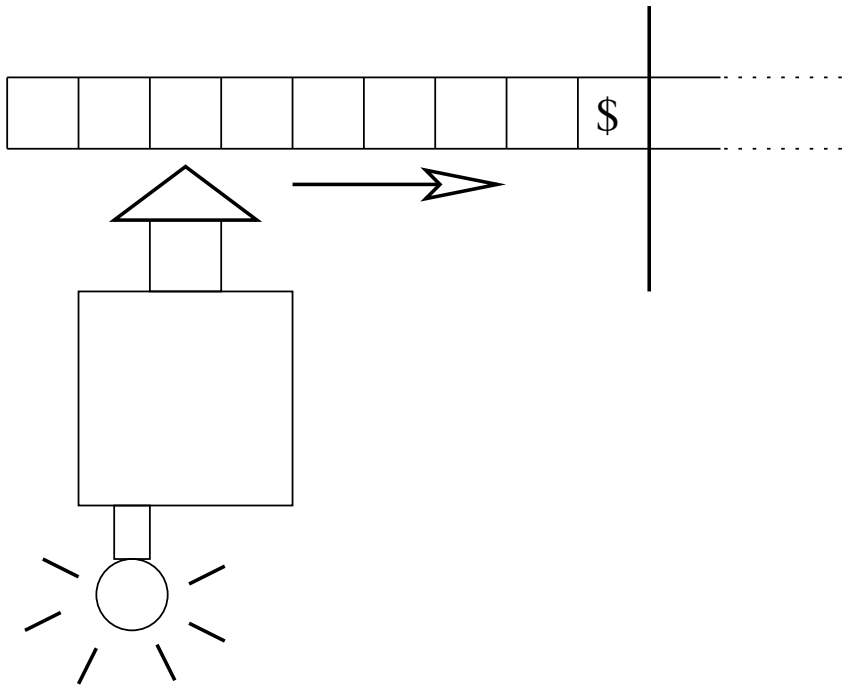
A rational language on Σ is a subset of Σ^* inductively defined thus:

- $\emptyset$ 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

1. Formal Languages
2. Formal Grammars
3. Regular Languages
   - Definition
   - Automata
   - Properties
4. Formal complexity of Natural Languages

Metaphoric definition



Formal definition

Def. 16 (Finite deterministic automaton (FDA))

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

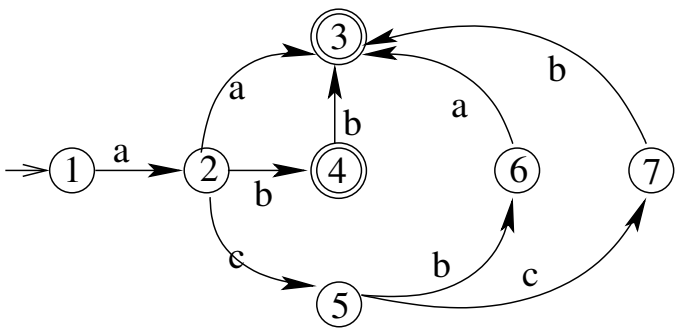
$$\mathcal{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$$

- 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:

δ : (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



	a	b	c
→ 1	2		
2	3	4	5
← 3			
← 4		3	
5		6	7
6	3		
7		3	

Sorbonne Nouvelle

Recognition

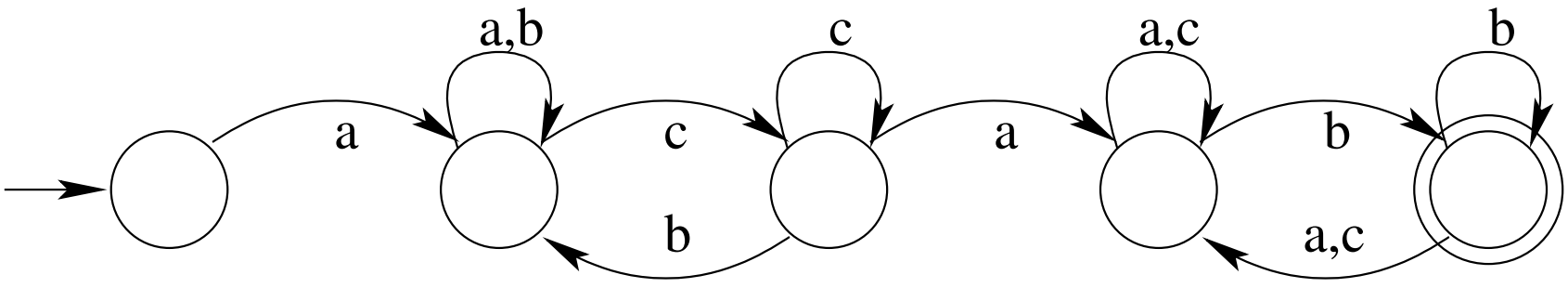
Recognition is defined as the existence of a sequence of states defined in the following way. Such a sequence is called a path in the automaton.

Def. 17 (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

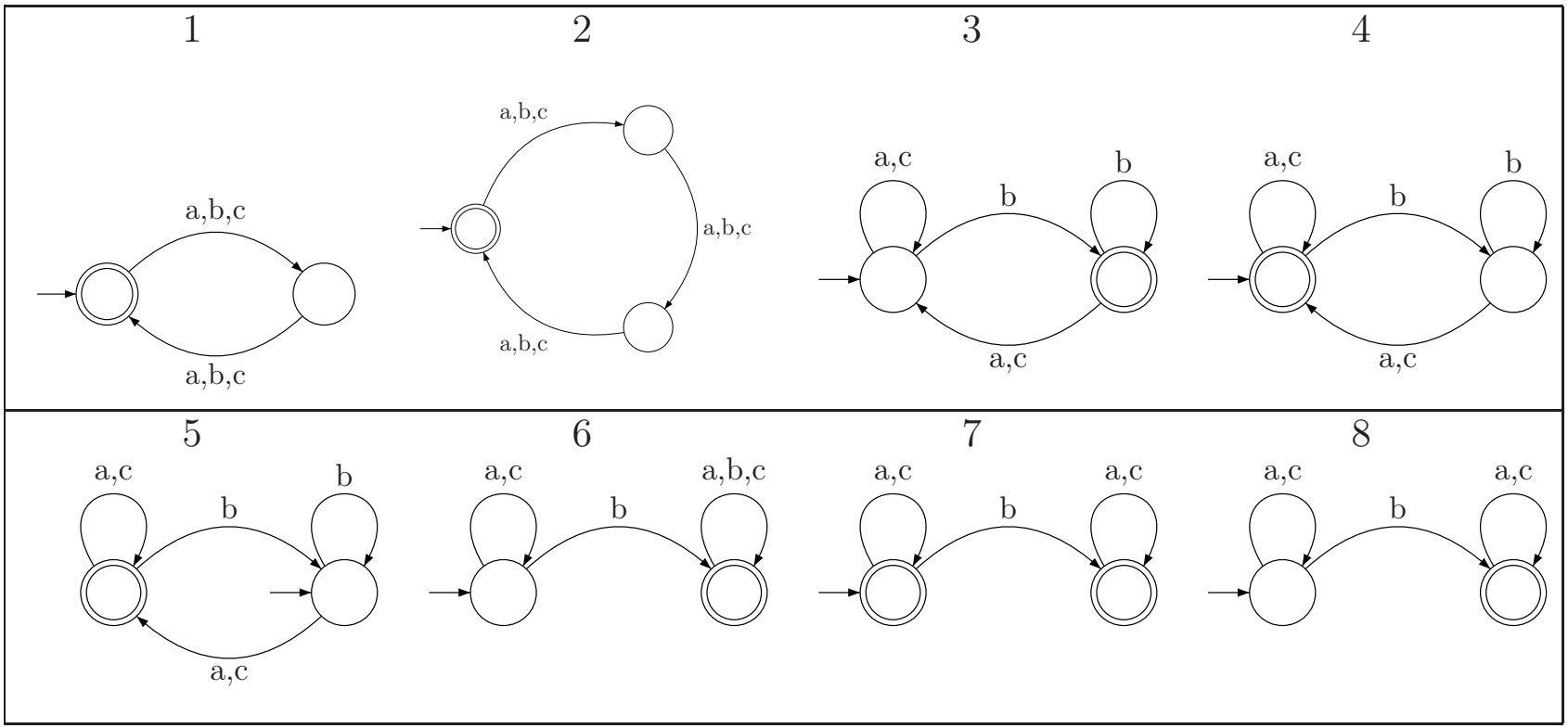


Exercices

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


1
a,b,c
a,b,c
2
a,b,c
a,b,c
a,b,c
3
a,c
b
b
a,c
4
a,c
b
b
a,c
5
a,c
b
b
a,c
6
a,c
b
a,b,c
7
a,c
b
a,c
8
a,c
b
a,c

Overview

1. Formal Languages
2. Formal Grammars
3. Regular Languages
   - Definition
   - Automata
   - Properties
4. Formal complexity of Natural Languages

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. 18 (Pumping Lemma)

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

$$\forall x \in L,\ |x| > k,\ \exists u, v, w \text{ such that } x = uvw, \text{ with:}$$
$$(i) \quad |v| \geq 1$$
$$(ii) \quad |uv| \leq k$$
$$(iii) \quad \forall i \geq 0,\ uv^i w \in L$$

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

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

to prove that $\mathcal{L}$ is

regular provide an automaton

not regular show that the pumping lemma does not apply

Pumping lemma: Consequences

Def. 19 (Consequences)

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

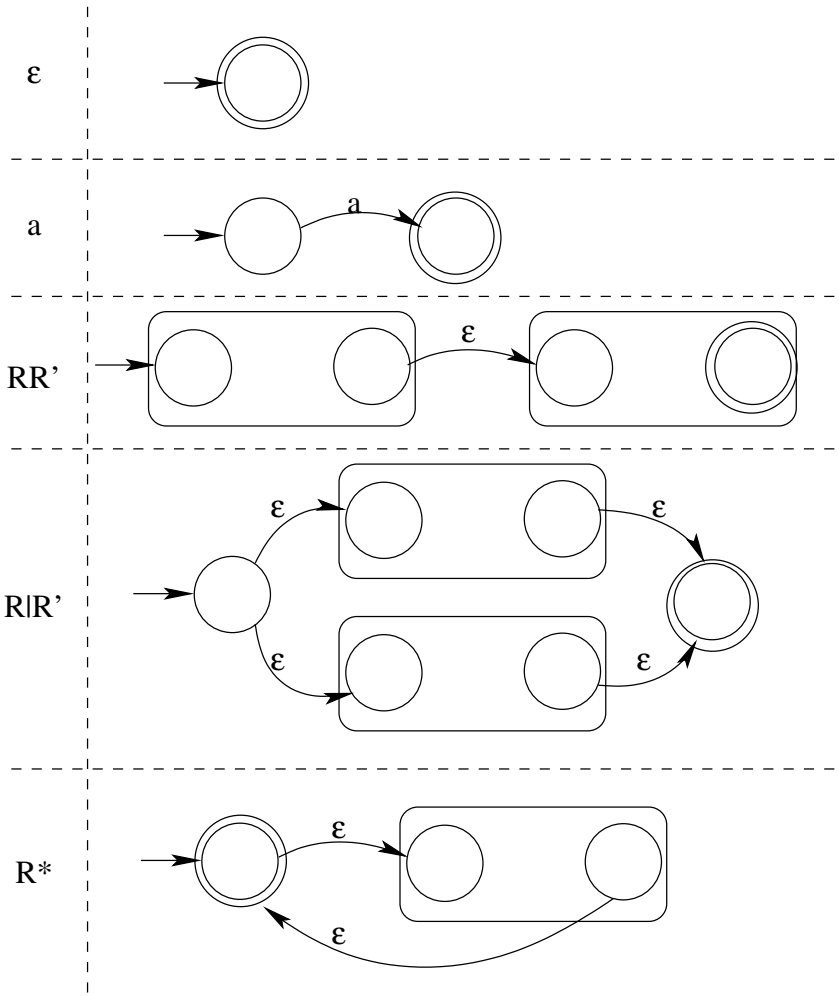
1. $L(\mathcal{A}) \neq \emptyset$ ***iff*** $\mathcal{A}$ recognises (at least) one word u s.t. $|u| < k$.
2. $L(\mathcal{A})$ is infinite ***iff*** $\mathcal{A}$ recognises (at least) one word u t.q. $k \leq |u| < 2k$.

Closure

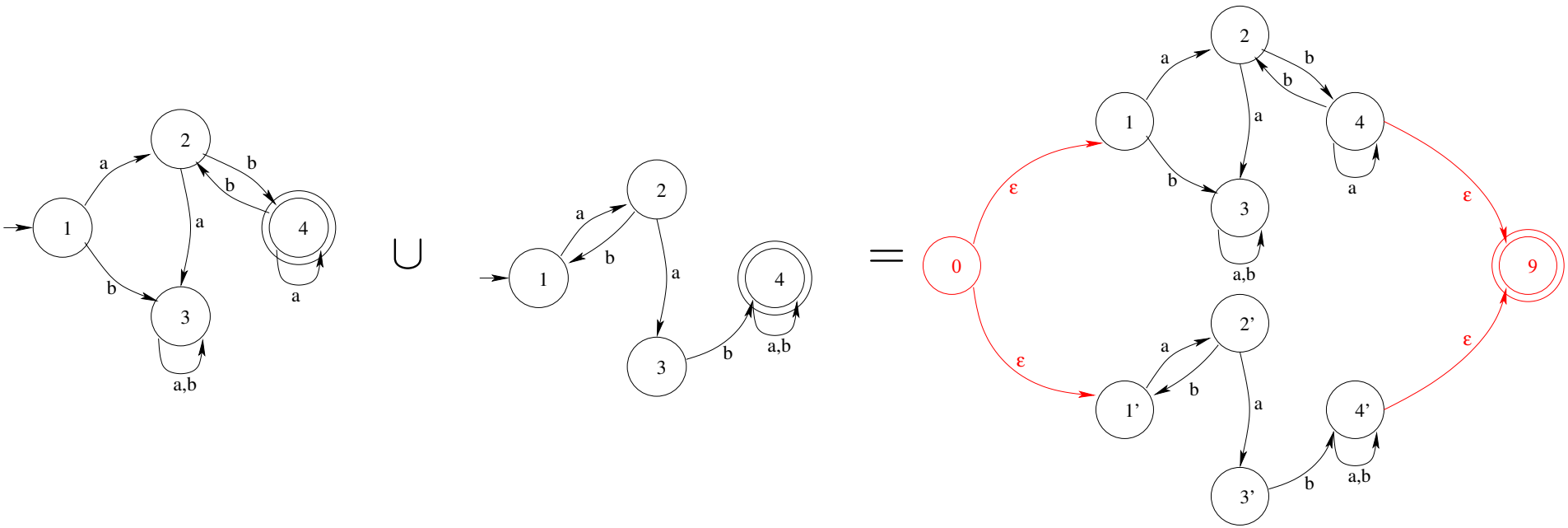
Regular languages are closed under various operations: if the languages L and L' are regular, so are:

- $L \cup L'$ (union); $L.L'$ (product); L^* (Kleene star)
  (rational operations)
- $L \cap L'$ (intersection); $\overline{L}$ (complement)
- ...

Rational operations



Union of regular languages: an example



Intersection of regular languages

Algorithmic proof

Deterministic complete automata

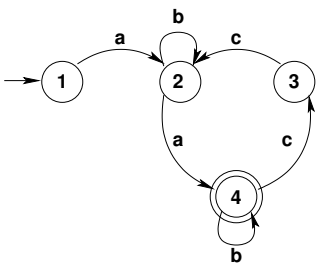
L_1	a	b
$\rightarrow$ 1	2	4
2	4	3
$\leftarrow$ 3	3	3
4	4	4

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

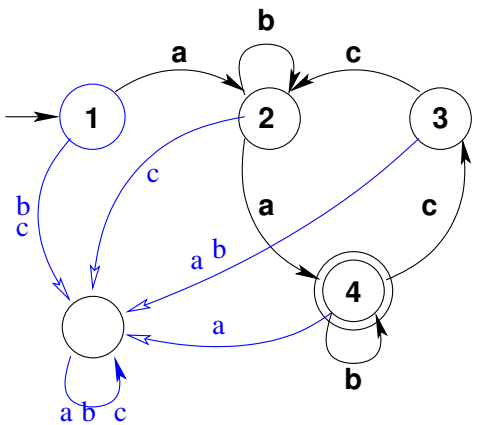
$L_1 \cap L_2$	a	b
$\rightarrow$ (1,1)	(2,2)	(4,5)
(2,2)	(4,5)	(3,3)
(4,5)	(4,5)	(4,5)
(3,3)	(3,4)	(3,5)
(3,4)	(3,1)	(3,4)
$\leftarrow$ (3,1)	(3,2)	(3,4)
(3,2)	(3,4)	(3,3)
(3,5)	(3,5)	(3,5)

Complement of a regular language

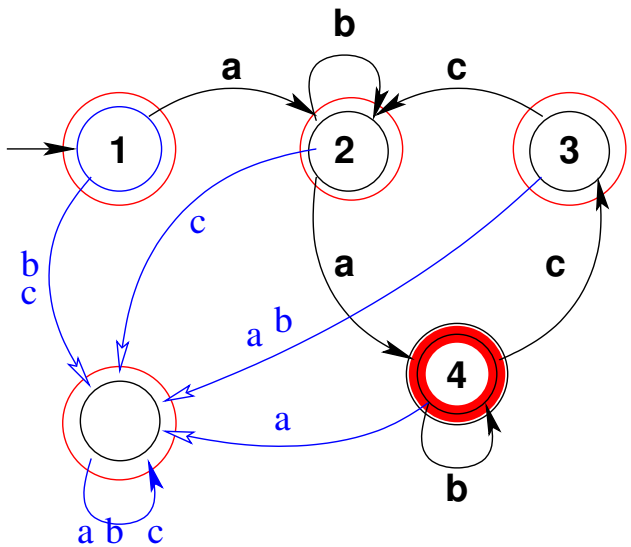
Deterministic complete automata



completed



complemented



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

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

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.

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" $w \stackrel{?}{\in} L(\mathcal{A})$ 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.

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

$\Rightarrow$ it boils down to answering the question:
$\left(L(\mathcal{A}) \cap \overline{L(\mathcal{A}')}\right) \cup \left(L(\mathcal{A}') \cap \overline{L(\mathcal{A})}\right) = \emptyset$

Sorbonne Nouvelle

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.

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.

Sorbonne Nouvelle