Formal Languages: An Introduction

Angelo Ortiz Tandazo

ENS-PSL & GIPSA-lab

Master's in Cognitive Science

Class information

Lectures: Tuesdays, 4 pm - 6 pm

Tutorials: Fridays, 4 pm - 6 pm

Official website: https://www.linguist.univ-paris-diderot.fr/~amsili/Ens/FTSL.php

Moodle site: https://moodle.psl.eu/enrol/index.php?id=25347

Please, enrol if you have not done so already!

Evaluation: 4 homeworks (60 %) and 1 final exam (40 %)

Contacts: pascal.amsili@ens.fr & angelo.ortiz.tandazo@ens.psl.eu

Outline

Motivation

Set theory refresher

Basic concepts

Finite automata

Exercises

Next steps

Motivation

- The child ate the pasta with a fork

- The child ate the pasta with a fork

- John saw the man on the mountain with a telescope.

Ambiguity

- The child ate the pasta with a fork
  - Did the pasta come with a fork?
  - Or did the child use a fork?
- John saw the man on the mountain with a telescope.
  - Was John on the mountain? Or was the man?
  - Did John had a telescope? Or did the man? Or did the mountain?

Motivation

- Your brain effortlessly resolves these ambiguities using context, world knowledge, and probability
- You can understand sentences you've never heard before
- You can detect grammatical errors even in unfamiliar contexts
    - ‘The ideas sleeps’ → Something feels wrong

Motivation

- ➢ Your brain effortlessly resolves these ambiguities using context, world knowledge, and probability
- ➢ You can understand sentences you've never heard before
- ➢ You can detect grammatical errors even in unfamiliar contexts
  - ‘The ideas sleeps’ → Something feels wrong

1. How does your mind represent grammatical rules?
2. What makes some sequences of words ‘feel’ right or wrong?
3. Can we create mathematical models that capture these intuitions?

Motivation

Formal languages will give us precise tools to:

- ➢ Model how our minds process structure
- ➢ Bridge human language processing and computational systems
- ➢ *Understand the mathematical limits of what patterns can be recognised*

Set theory refresher

Set theory refresher I

Set: Unordered sequence of unique elements

Empty set: $\varnothing$ or {}

Membership: $x \in S$ — 'x is an element/member of S'

Extensional notation: $\{x_1, x_2, \ldots, x_n\}$ — 'elements $x_1, x_2, \ldots, x_n$'

Intensional notation: $\{x \in S \mid P(x)\}$ — 'elements of S that verify property P'

Inclusion: $A \subseteq B$ — 'all the members of A are also members of B'

Cardinal: $|S|$ — '*number* of elements of S'

Set theory refresher II

Union: $A \cup B$ — ‘all the elements of A and B’

Intersection: $A \cap B$ — ‘all the common elements of A and B’

Complementation: $A \setminus B$ — ‘all the elements of A not in B’

Power set: $\mathscr{P}(S)$ — ‘all the combinations of elements of S’

<u>ex:</u> $\mathscr{P}(\{0, 1\}) = \{\varnothing, \{0\}, \{1\}, \{0, 1\}\}$

Set theory refresher III

$\mathbb{N}^* = \{x \in \mathbb{N} \mid x \geq 1\} \subseteq \mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{R}$

$|\varnothing| = 0$

For any set S:

$\varnothing \subseteq S$ $\quad$ $S \subseteq S$

$S \cup \varnothing = S$ $\quad$ $S \cap \varnothing = \varnothing$

$\varnothing \in \mathscr{P}(S)$ $\quad$ $S \in \mathscr{P}(S)$

Basic concepts

Alphabet

➢ A finite set of symbols denoted by Σ.

Examples:

- English alphabet: $\Sigma_{en} = \{a, b, c, ..., z\}$
- Binary alphabet: $\Sigma_{bin} = \{0, 1\}$
- Programming alphabet: Σ_{prog} = {letters, digits, operators, brackets, ...}

Word

➢ Any finite sequence (i.e. string) of symbols (i.e. letters) from the alphabet.

Examples:

- ε (empty word)
- ‘hello’ is a valid English word
- ‘101’ is a string in binary
- ‘x = y + 2’ is a string in most programming languages

Length operator

➢ For any word w, $|w|$ denotes the number of symbols in w.

Examples:

- $|\varepsilon| = 0$
- |‘hello’| = 5
- |‘101’| = 3

Concatenation

➢ For any words u and v, its concatenation uv (or $u.v$) is the word formed by adding v at the end of u.

Notation: $\underbrace{u.u.u\ldots u}_{k\ times} = u^k$

Special sets

- $\Sigma^k \triangleq \{w \mid |w| = k \text{ and } w \text{ is a string over } \Sigma\}$, for $k \geq 1$
- $\Sigma^0 \triangleq \{\varepsilon\}$, regardless of Σ
- $\Sigma^1 = \Sigma$
- $\Sigma^* \triangleq \bigcup_{k \geq 0} \Sigma^k$

Note: if $\Sigma = \varnothing$, then $\Sigma^* = \{\varepsilon\}$.

Examples: for $\Sigma_{bin} = \{0, 1\}$

- $\Sigma_{bin}^1 = \{0, 1\}$
- $\Sigma_{bin}^2 = \{00, 01, 10, 11\}$
- $\Sigma_{bin}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

Language

➢ A set of words over an alphabet Σ, that is a subset of Σ^*.

Examples:

- English *dictionary*: all valid English words
- Binary strings: all sequences of 0s and 1s
- Binary numbers: all sequences of 0s and 1s without leading 0s
- Python: all syntactically valid Python programs

Letters, words and language

Language: English *dictionary*

Alphabet?

Letters, words and language

Language: English *dictionary*

Alphabet? **English morphemes**

Letters, words and language

Language: English *dictionary*

Language: English *sentences*

Alphabet? English morphemes

Alphabet?

Letters, words and language

Language: English *dictionary*

Language: English *sentences*

Alphabet? English morphemes

Alphabet? **English words**

Letters, words and language

Language: English *dictionary*

Alphabet? English morphemes

Language: English *sentences*

Alphabet? English words

Language: Morse code

Alphabet?

Letters, words and language

Language: English *dictionary*

Language: English *sentences*

Language: Morse code

Alphabet? English morphemes

Alphabet? English words

Alphabet? **{., –}**

Finite automata

Finite automata

➢ An abstract model of computation that, starting from an **initial state**, reads an input (i.e. a word) from left to right and accepts iff it ends up in an **accept state** after reading the whole input.

Elements:

- Q: a finite set of states
- Σ: an alphabet
- q_0: the initial state
- $F \subseteq Q$: final states
- δ: a transition function

Finite automata

- ➢ An abstract model of computation that, starting from an **initial state**, reads an input (i.e. a word) from left to right and accepts iff it ends up in an **accept state** after reading the whole input.
- ➢ A finite automaton A recognises a language L iff its set of accepted words $\mathscr{L}(A) = L$.

Elements:

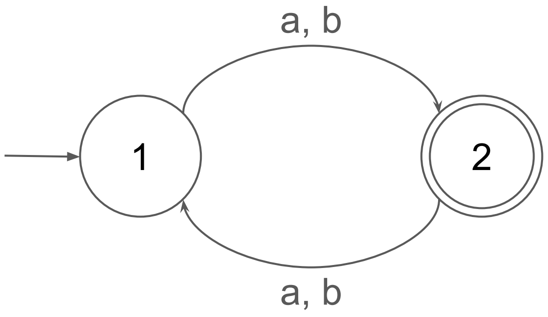
- Q: a finite set of states
- Σ: an alphabet
- q_0: the initial state
- $F \subseteq Q$: final states
- δ: a transition function

Finite automata: example

State diagram of a finite automaton recognising the set of words with an odd length over the alphabet $\Sigma = \{a, b\}$.

Finite automata: example

State diagram of a finite automaton recognising the set of words with an odd length over the alphabet $\Sigma = \{a, b\}$.

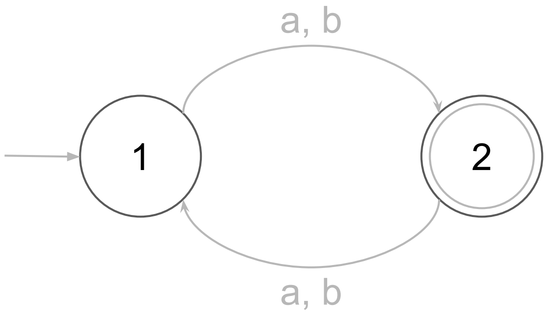


Finite automata: example

State diagram of a finite automaton recognising the set of words with an odd length over the alphabet Σ = {a, b}.

Elements:

- Q: a finite set of states
- Σ: an alphabet
- q_0: the initial state
- $F \subseteq Q$: final states
- δ: a transition function

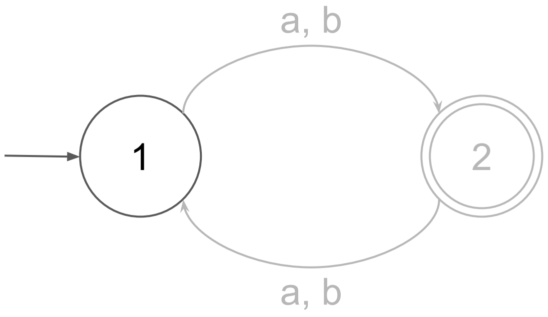


Finite automata: example

State diagram of a finite automaton recognising the set of words with an odd length over the alphabet Σ = {a, b}.

Elements:

- Q: a finite set of states
- Σ: an alphabet
- q_0: the initial state
- $F \subseteq Q$: final states
- δ: a transition function

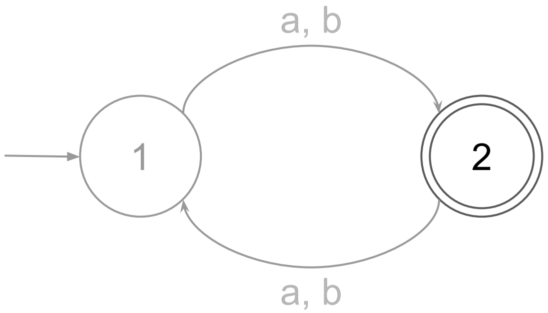


Finite automata: example

State diagram of a finite automaton recognising the set of words with an odd length over the alphabet Σ = {a, b}.

Elements:

- Q: a finite set of states
- Σ: an alphabet
- q_0: the initial state
- $F \subseteq Q$: final states
- δ: a transition function

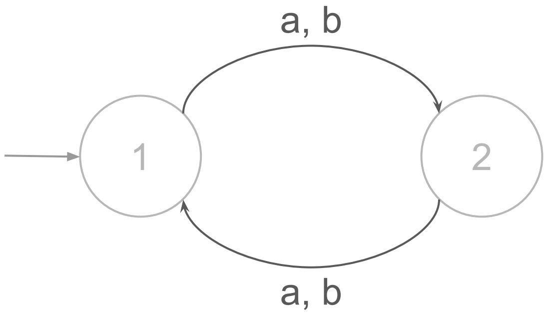


Finite automata: example

State diagram of a finite automaton recognising the set of words with an odd length over the alphabet Σ = {a, b}.

Elements:

- Q: a finite set of states
- Σ: an alphabet
- q_0: the initial state
- $F \subseteq Q$: final states
- δ: a transition function



Exercises!

Next steps

- Regular languages
- Formal grammars
- Complexity hierarchy
- First-order logic
- Predicate logic
- (General) quantifiers
- λ-calculus