# Automata

## course based on Jurafsky and Martin [2009, Chap.2]

UPPSALA
UNIVERSITET
Marie Dubremetz
marie.dubremetz@lingfil.uu.se

Uppsala, 2016

# Presentation Plan

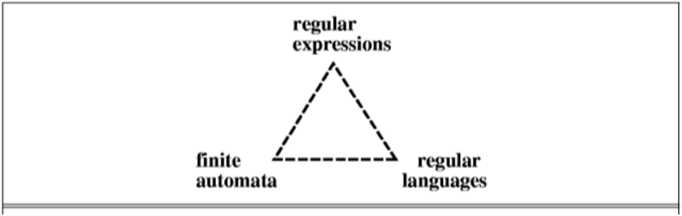
1. **Automata**
   - Definition
   - Exercise 1 on Formalism
   - Exercise 2 on Regex
2. **Determinism and Automata**
   - Definition
   - Example
3. **Summary**

# Table of Contents

# Table of Contents

# Why Automata ?

Every regular expression are based on a formalism called automata. Without knowing it, you apply it all the time. It is a basic formalism on which you will build your knowledge to understand other formalisms.



# Definition

**Finite-state automaton** (FSA, plural : automata), or simply a state machine, is a mathematical model of computation used to design computer programs and regular expressions. It is conceived as an abstract machine that can be in one of a finite number of **states**.

The machine is in only one state at a time ; the state it is in at any given time is called the **current state**. It can change from one state to another when initiated by a triggering event or condition ; this is called a **transition**.

A particular FSM is defined by a list of its states, and the triggering condition for each transition. **Deterministic finite automaton** (DFA) and **nondeterministic finite automaton** (NFA) are subcategories of Finite-state automaton

# Definition

An automaton can be described by its formal definition. Here is the lexicon of FSA formalism :

- $Q$: a finite set of $N$ states $q_0, q_1, \ldots q_N$
- $\Sigma$: a finite input alphabet of symbols
- $q_0$: the start state
- $F$: the set of final states, $F \subseteq Q$
- $\delta(q,i)$: the transition function or transition matrix between states. Given a state $q \in Q$ and an input symbol $i \in \Sigma$, $\delta(q,i)$ returns a new state $q' \in Q$. $\delta$ is thus a relation from $Q \times \Sigma$ to $Q$;

# Definition

An automaton can as well get a visual representation

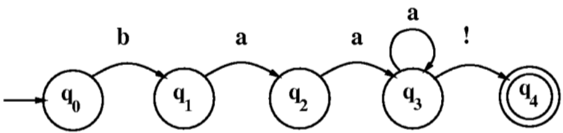



**Figure 2.10** A finite-state automaton for talking sheep.

For the sheeptalk automaton in Figure 2.10, $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{a, b, !\}$, $F = \{q_4\}$

# Table of Contents

# Exercise 1 on Formalism

The next slide describes an Automaton
According to the formalism, color/circle what represents :

- Q in blue
- $\Sigma$ in green
- $q_0$ and $F$ (if not missing) in dotted blue
- $\delta$ (q,i) in red

- $Q$: a finite set of $N$ states $q_0, q_1, \ldots q_N$
- $\Sigma$: a finite input alphabet of symbols
- $q_0$: the start state
- $F$: the set of final states, $F \subseteq Q$
- $\delta(q,i)$: the transition function or transition matrix between states. Given a state $q \in Q$ and an input symbol $i \in \Sigma$, $\delta(q,i)$ returns a new state $q' \in Q$. $\delta$ is thus a relation from $Q \times \Sigma$ to $Q$;

# Exercise

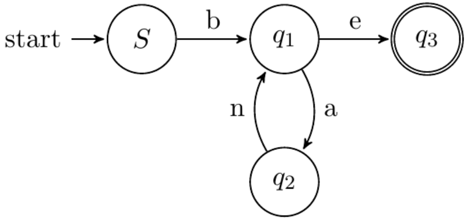



Figure 1: Finite State Automaton, accepting the pattern $b(an)+e$

# Table of Contents

# Exercise 2 on Regex

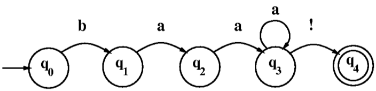



**Figure 2.10** A finite-state automaton for talking sheep.

## Instructions 1

ba

baaa !

baaa

baa !

b

baaaaaaaaaaaaa !

- Which of those examples should be accepted by this automaton ?

# Exercise 2

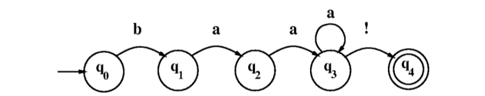



**Figure 2.10** A finite-state automaton for talking sheep.

## Instructions

ba

baaa !

baaa

baa !

b

baaaaaaaaaaaaa !

- Open a terminal window and find the grep regular expression that correspond to the image up. Apply it to the file, do you get the right output ?

# Exercise 2 on Regex

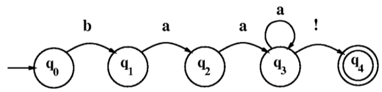



**Figure 2.10** A finite-state automaton for talking sheep.

## Conclusion on the exercises

Congratulations you went all the way from formalism to implementation !

# Table of Contents

# Table of Contents

# Definition

In theory of computation a **deterministic finite automaton (DFA)**—also known as deterministic finite accepter (DFA)is a finite state machine that accepts/rejects finite strings of symbols and which transitions is determined only by the current state and the symbol it is looking at.
If in an automaton a state can lead to two ways with the same symbol this automaton is called non-deterministic finite automaton.

# Table of Contents

# Example Deterministic VS Non-deterministic

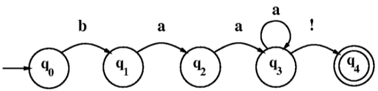



**Figure 2.10** A finite-state automaton for talking sheep.

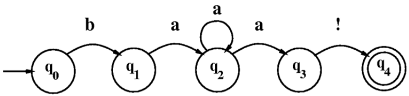



**Figure 2.18** A non-deterministic finite-state automaton for talking sheep (NFSA #1). Compare with the deterministic automaton in Figure 2.10.

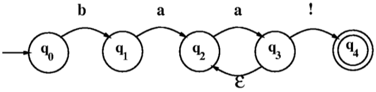



**Figure 2.19** Another NFSA for the sheep language (NFSA #2). It differs from NFSA #1 in Figure 2.18 in having an ε-transition.

The non-determinism need to be handled by algorithms like Back-up, look-ahead and parallelism in order avoid errors.

# Table of Contents

# Summary

- We saw a formalism called **automaton**
- Automaton is the basis of several other formalism and of Regex
- Automata can be **deterministic** or **non-deterministic**
- They are a way to represent the step of a reasoning in the most computational friendly way.
- To be academic, you must always check that all the element of the formalism is present in the automaton representation you create (Start, Final states, symboles etc.)

# References

Daniel Jurafsky and James H Martin. *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 163 of *Prentice Hall Series in Artificial Intelligence*. Prentice Hall, 2009.