Automata Theory

How Machines Hear and Recognize Speech

Emily Chen, Arya Nayak, Hazel Thrasher

Mentor: Alicia Lin

Overview

1. Finite Automata
2. Context-Free Grammars and Pushdown Automata
3. Applications of Automata Theory
4. Conclusion

So, what is a computer?

FINITE AUTOMATA

1. Limited memory, useful for fundamental tasks
2. Visualization: state diagrams
3. Start state, accept state, transitions
4. Accept vs. Reject

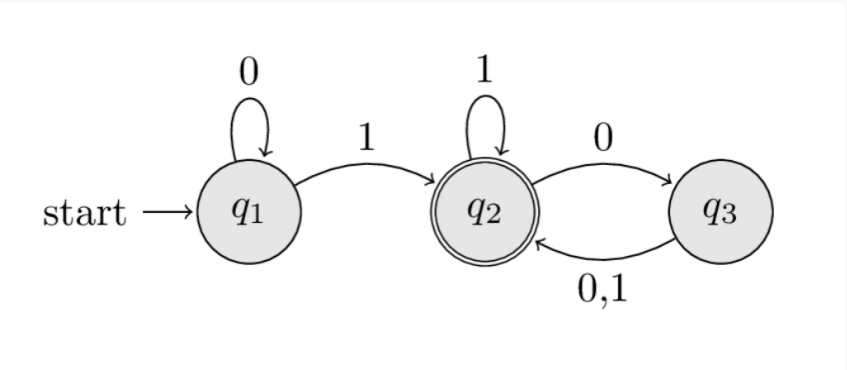


Figure 1: State diagram of a finite automaton with start state q_1 and accept state q_2

Formal Definition of a Finite Automaton

A **finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q represents the finite set of states,
2. Σ represents the input alphabet,
3. $\delta : Q \times \Sigma \to Q$ is the transition function,
5. $q_0 \in Q$ is the start state, and
6. $F \subseteq Q$ is the set of accept states.

Nondeterminism

- DFAs vs. NFAs: Similarities and Differences
- Several choices of which step to take next
- ε - nondeterministic decisions

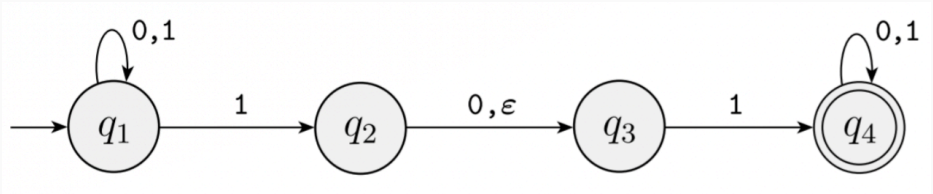


Figure 2: State diagram of NFA with start state q_1 and accept state q_4

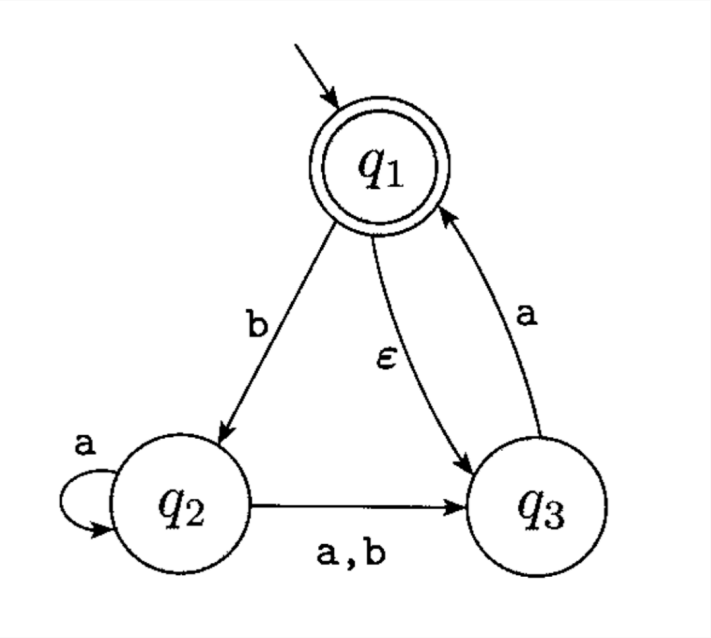


Figure 3:

State diagram of a NFA with start and accept state q_1

Regular Languages and Operations

1. A language L is a **regular language** if there exists a finite automaton that accepts exactly the strings in L.

2. Regular Operations

   - **Union:** $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
   - **Concatenation:** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
   - **Star:** $A* = \{x_1 x_2 ... x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

The class of regular languages is **closed** under the operations of union, concatenation, and star.

REGULAR EXPRESSIONS

- Arithmetic expressions use $+$, $-$, $\times$, $\div$
- **Regular** expressions use $\cup$, $\circ$, $*$
- While an arithmetic expression usually yields a number, a regular expression yields a regular language.
- Arithmetic expression: $(5 + 7) \times 3 + 6 \div 2 = 39$
- Regular expression: $(0 \cup 1)1*$
- Order of Operations!

Finite Automata $\leftrightarrow$ Regular Expressions

- Regular expressions may seem very different from finite automata
- They are actually equivalent in their descriptive power

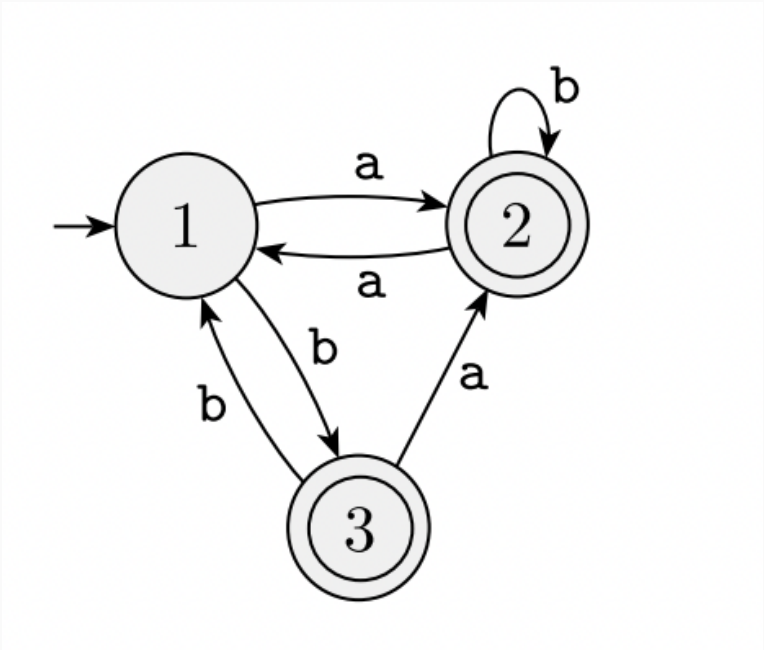


Figure 4: State diagram of a DFA

The equivalent (very long) regular expression!

$$(a(aa \cup b) * ab \cup b)((ba \cup a)(aa \cup b) * ab \cup bb) * ((ba \cup a)(aa \cup b) * \cup \varepsilon) \cup a(aa \cup b) *$$

Context-Free Grammars and Pushdown Automata

Introduction to Context-Free Grammars

- Finite automata have **limitations**
- CFGs describe **recursive** structures
- Useful in real-world applications
  - programming languages
  - parsers
  - speech recognition

$$L(G_1) = \{0^n \# 1^n \mid n \geq 0\}$$

Examples:

- $\#$
- $0\#1$
- $00\#11$
- $000\#111$

How CFGs Generate Strings

Rules:

- $A \to 0A1$
- $A \to B$
- $B \to \#$

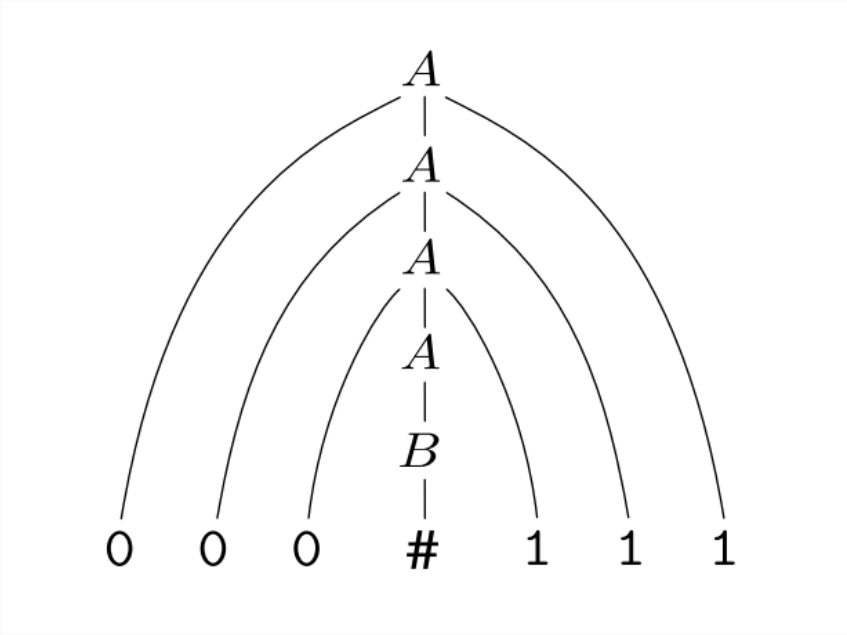


Figure 5: Parse tree for 000#111 in grammar G_1

Derivation:

$$A \Longrightarrow 0A1 \Longrightarrow 00A11 \Longrightarrow 000A111 \Longrightarrow 000B111 \Longrightarrow 000\#111.$$

Formal Definition of CFGs + Context-Free Languages

$$(V, \Sigma, R, S)$$

Table 1: Components of a Context-Free Grammar

Symbol	Meaning
V	variables
Σ	terminals
R	rules
S	start variable

$$L(G_1) = \{0^n \# 1^n \mid n \geq 0\}$$

Introduction to Pushdown Automata

- PDA = finite automaton + **stack**
- Stack provides **memory**
- Recognizes some nonregular languages

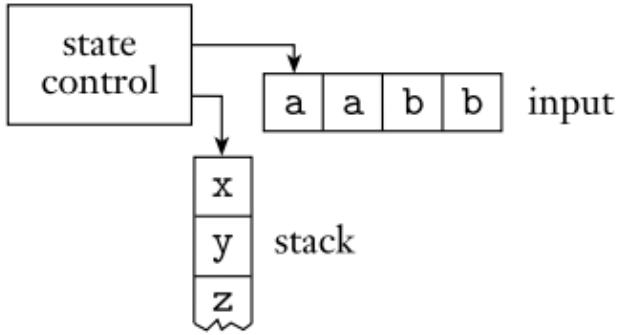


Figure 6: Schematic of a pushdown automaton

An example of a language recognizable by a PDA

$$\{0^n 1^n \mid n \geq 0\}$$

How a PDA Works

Example for recognizing $\{0^n 1^n \mid n \geq 0\}$

1. Read 0 → push onto stack
2. Read 1 → pop from stack
3. Accept if input ends and stack is empty

Input: 0011

1. Read 0 → Stack: 0
2. Read 0 → Stack: 00
3. Read 1 → Stack: 0
4. Read 1 → Stack: ε

Formal Definition of PDAs

$$(Q, \Sigma, \Gamma, \delta, q_0, F)$$

Table 2: Components of a Pushdown Automata

Symbol	Meaning
Q	states
Σ	input alphabet
Γ	stack alphabet
δ	transition function
q_0	start state
F	accept states

$$\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$$

Nondeterministic PDA Example

Language:

$$\{a^i b^j c^k | i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$$

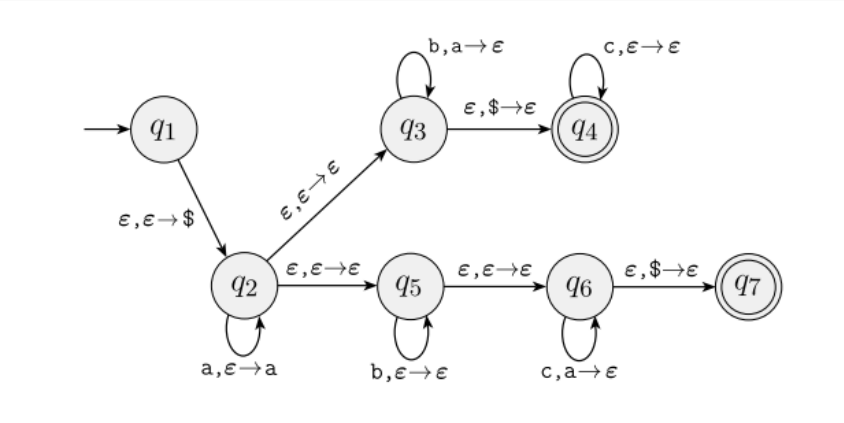
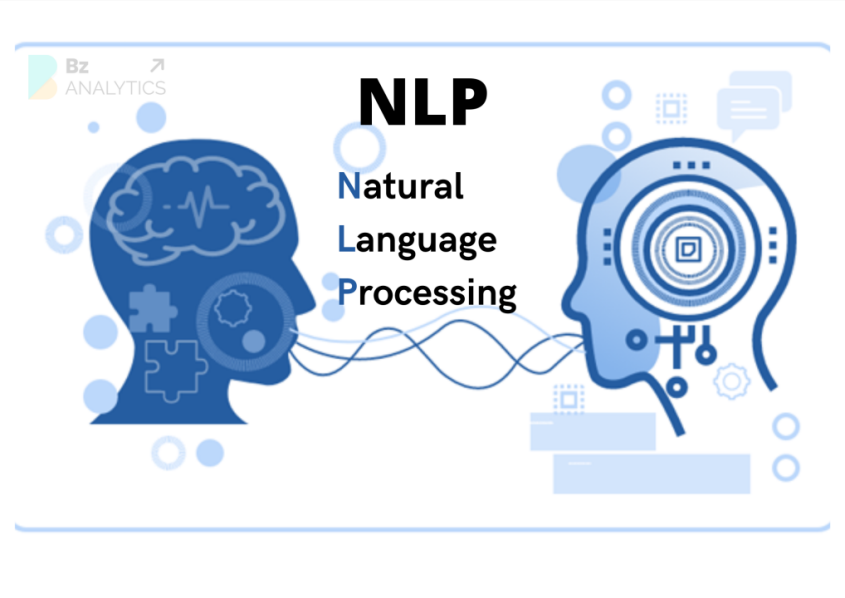


Figure 7: State diagram for PDA M_1 that recognizes

- PDA "guesses"
- Matches a's with:
  - b's OR
  - c's
- Accept if one branch succeeds

Why Is This Important?

Applications of CFGs in NLP

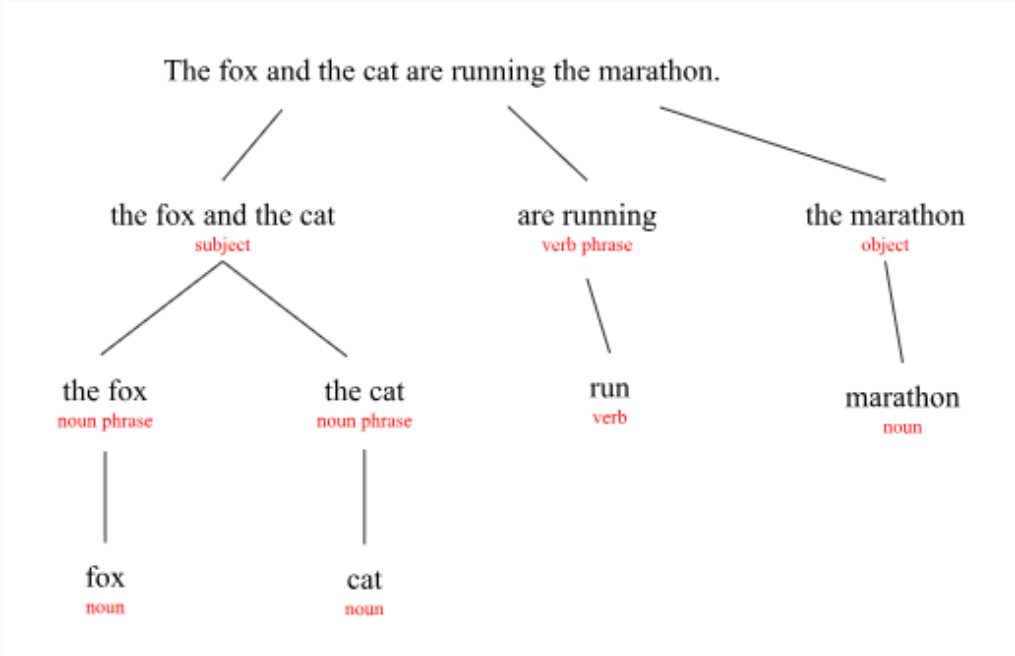


- NLP = Natural Language Processing
- One real-world application of Automata Theory (Context-Free Grammars)
- Allows machines to understand, interpret, and generate language

Tokenization

CFGs can perform **TOKENIZATION**, the process of breaking down sentences and words into smaller, processable parts:

- Subject/verb/object
- Noun phrases and verb phrases
- Nouns, verbs, adjectives and adverbs
- Affixes, roots and suffixes



How CFGs help with NLP

- Variables/non-terminal symbols = parts of speech
- Terminals = words

[SENTENCE]	→	[NOUN-PHRASE][VERB-PHRASE]
[NOUN-PHRASE]	→	[CMPLX-NOUN] | [CMPLX-NOUN][PREP-PHRASE]
[VERB-PHRASE]	→	[CMPLX-VERB] | [CMPLX-VERB][PREP-PHRASE]
[PREP-PHRASE]	→	[PREP][CMPLX-NOUN]
[CMPLX-NOUN]	→	[ARTICLE][NOUN]
[CMPLX-VERB]	→	[VERB] | [VERB][NOUN-PHRASE]
[ARTICLE]	→	a | the
[NOUN]	→	child | parent | dog
[VERB]	→	touches | helps | pets
[PREP]	→	with

Figure 8: Example of how a CFG might generate a sentence

Applications of NLP

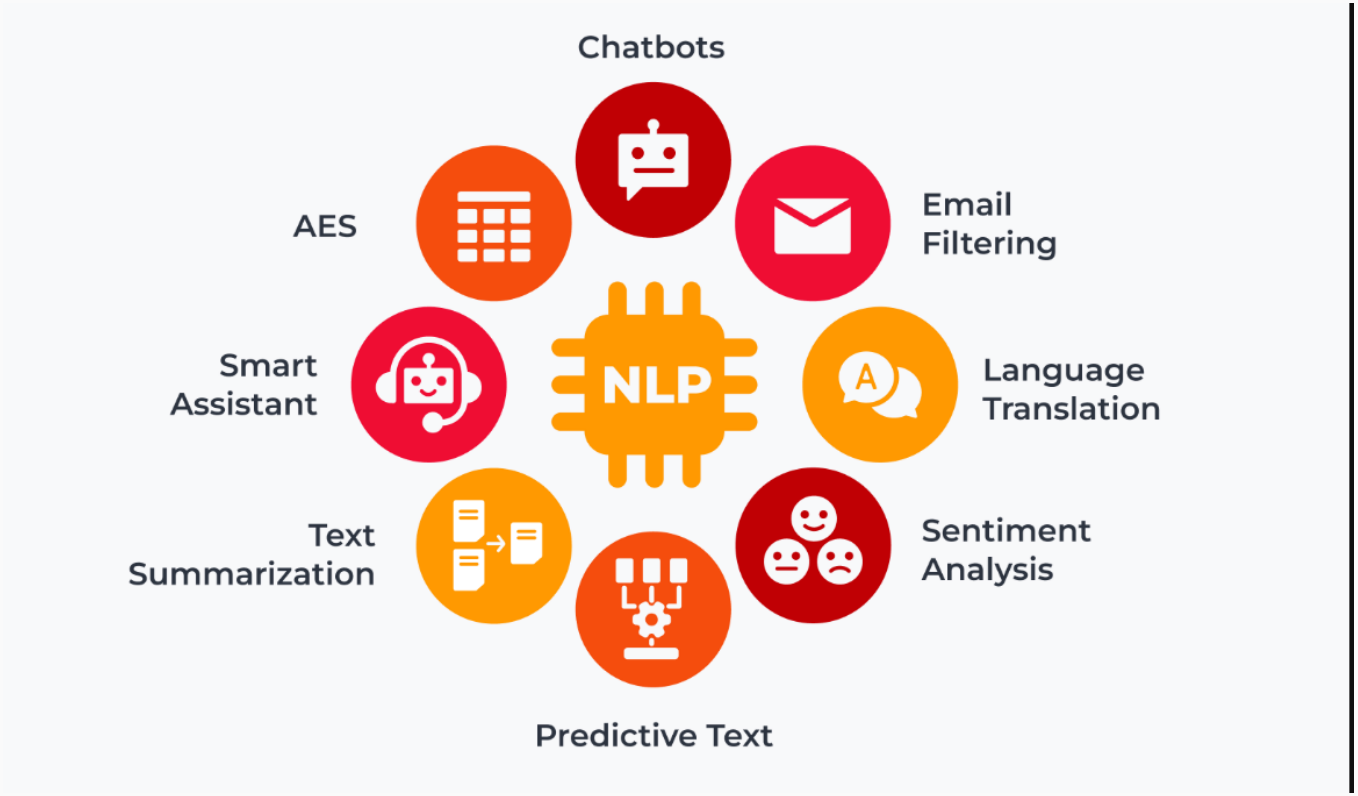


Figure 9: Some possible applications of NLP

General Applications of Automata Theory

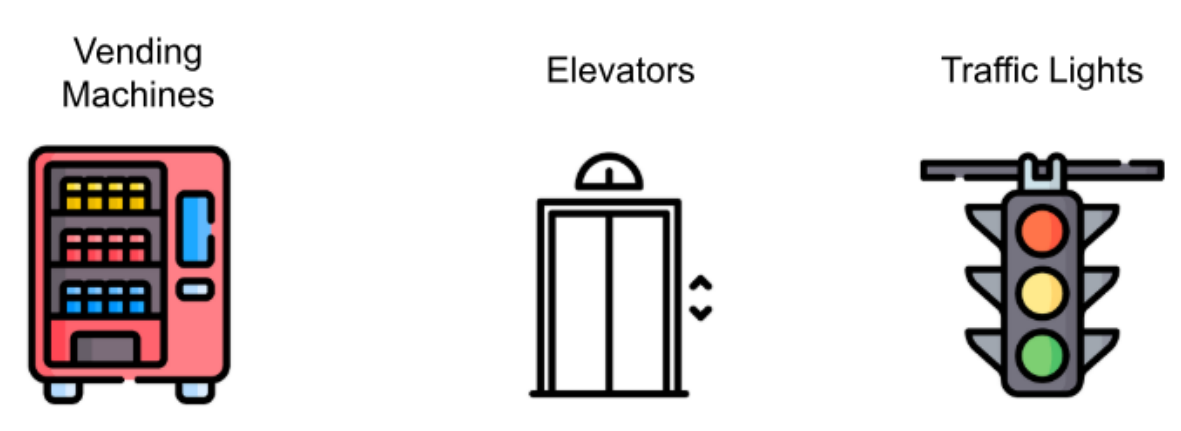


Figure 10: Examples of some of the ways Finite Automata can be used in the real world

CFLs and Automata Theory help mathematicians understand the capabilities of real-world computers:

- What problems they can and cannot solve
- The space & time needed to solve them

Conclusion

Summary

Finite Automata $\rightarrow$ limited memory and capabilities.

Pushdown Automata $\rightarrow$ more memory because of its access to a **stack**

Context-Free Grammars (CFGs) **generate** languages while Pushdown Automata (PDAs) **recognize** them.

CFGs can model Natural Language Processing and speech recognition.

Thank You!

References i

[1] Sipser, M. (2013). Introduction to the Theory of Computation (3rd ed.).