Theory of Computation (1)

Coding Club

Mohammed Alshamsi
2021004826
mo.alshamsi@aurak.ac.ae

American University of Ras Al Khaimah

April 14, 2024

Introduction

- **Theory of Computation?** Study of theoretical models of computation.
- **Format?** You'll see the following today:
  - Mathematical definitions with examples
  - Applications to compilers
- **Why?** Modelling behavior and processing strings are major applications of this area.
- **These Slides?** See our GitHub repository. (More topics and applications also available.)
- **If you don't know coding:** Check document on GitHub, or find a guide online.

Any questions?

Outline

Alphabets and Strings

Definition (Alphabet)

An alphabet Σ is any nonempty finite set. The members of the alphabet are said to be the *symbols* of the alphabet.

Definition (Strings)

A string w over Σ is a finite sequence of symbols from Σ.

ϵ is the empty string.

wv is the concatenation of two strings w, v.

w^n is concatenation of n copies of w. Also, $w^0 = \epsilon$.

Languages

Definition (Languages)

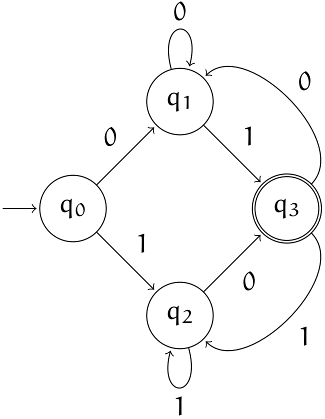
Language: set of strings over an alphabet.

Empty language $\emptyset$ contains no strings.

Kleene closure Σ^* of an alphabet Σ is the set of all strings over Σ.

Deterministic Finite Automata

Start at q_0 and follow the arrows according to an input string of 0's and 1's.
q_3 is a final state. If input ends at q_3, string is accepted.



Deterministic Finite Automata

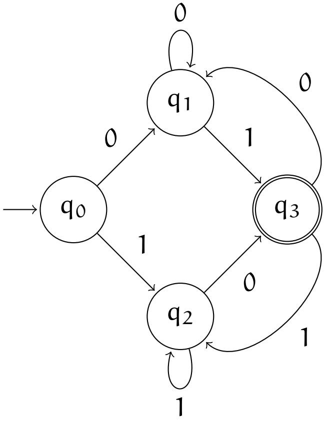
DFAs: Easy to implement! That one took 30 lines.

Definition (Deterministic Finite Automaton)

A DFA has five components:

1. Q is a set of states
2. $q_0 \in Q$ is the initial state
3. $F \subseteq Q$ is the set of final states
4. Σ is the input alphabet
5. δ is the transition function

Example



1. $Q = \{q_0, q_1, q_2, q_3\}$
2. The initial state is q_0.
3. The set of final states is $\{q_3\}$.
4. The input alphabet Σ is $\{0, 1\}$.
5. δ represented by this table:

δ	0	1
q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_3	q_2
q_3	q_1	q_2

Operations on Languages

Definition

Given two languages A and B:

- Union: $A \cup B$ contains all strings in A and all strings in B.
- Concatenation: AB contains all strings ab, where $a \in A$ and $b \in B$.
  - A^n is concatenation of A with itself n times.
- Closure: A^* is union of A^0, A^1, A^2, $\ldots$

Regular Languages

- Regular languages and DFAs are equivalent!
  - Any regular language has some DFA that recognizes it.
  - Any language recognized by a DFA is regular.
- But what are regular languages?

Regular Languages

Definition (Regular Language over Σ)

Recursive definition.

- $\emptyset$ is regular
- For every $a \in \Sigma$, $\{a\}$ is regular
- If A is regular, A^* is regular
- If A, B are regular, $A \cup B$ and AB are regular
- No other languages over Σ are regular

Regular Expressions

Notation for expressing regular languages.

Example (Regular Expressions)

$\mathbf{ab} + \mathbf{a}^*$ is union of $\mathbf{ab}$ and $\mathbf{a}^*$

$\mathbf{ab}$ is concatenation of $\mathbf{a}$ and $\mathbf{b}$

$\mathbf{a}^*$ is closure of $\mathbf{a}$

Applications to Compilers

- Traditional compiler design:
  1. Lexical Analysis (or Tokenization)
  2. Syntax Analysis
  3. Semantic Analysis
  4. Code Generation
- Regular languages help with tokenization.
- Common tool is `Flex`. Input is regular expression, output is tokenizer.

Demo: Calculator

We'll make a basic calculator "language" that has:

- `let` keyword to declare variables
- Identifiers for the variables (letters only)
- Numbers, floating-point only
- `+`, `-`, `*`, `/` for arithmetic
- `(` and `)` for grouping
- `=` for assignment

Paradigm: Input a statement and press enter. If it can be evaluated (i.e. not declaration or assignment), result will be printed.

Flex Regex Syntax

- `ab` matches `a` followed by `b`
- `a|b` matches "either `a` or `b`"
  - `[ab]` is equivalent
  - `[a-z]` is shorthand for `a|b|...|z`
  - `[A-Z]` is `A|B|...|Z`
  - `[0-9]` is `0|1|...|9`
- `a*` matches "zero or more consecutive occurrences of `a`"
- `a+` matches "one or more consecutive occurrences of `a`"
- `.` matches any character

That's All!

Most of this was based on the following:

- Compilers: Principles, Techniques, and Tools (Aho et al.) — Chapter 3
- Introduction to the Theory of Computation (Sipser) — Chapters 0, 1