CSE 555 Theory of Computation

Goran Konjevod

Department of Computer Science and Engineering
Arizona State University

ASU, Spring 2008

Logistics 1

- Instructor: Goran Konjevod, BY450, goran@asu.edu
- Office Hours: TTh 10:30–12:00
- TA: Wei Chen, BY415AA, wchen10@asu.edu
- Office Hours: M 12:30–2:00, W 4:30–6:00

Logistics 2: Evaluation

- Homeworks (7–9), 30%
- Midterm (weekend take home), 30%
- Final (Thu 5/1, 7:40–9:30AM), 40%
- Programming Assignment (3/6–4/29), 10%

Homework 1

Due 1/22 at the **beginning** of class.

Homework 1

Due 1/22 at the **beginning** of class, at 9:15AM.

Homework 1

Due 1/22 at the **beginning** of class, at 9:15AM.
Late homework will not be accepted.

Outline of Course

1. Regular languages
2. Context-free languages
3. Turing machines and decidability
4. Complexity theory

General Questions

What is Computation?
How to think about Computation?
Better: How to model Computation mathematically?

Computation: basic properties

Input $\longrightarrow$(process)$\longrightarrow$ Output.
To describe a computation, must describe formally

1. Input
2. Computation process
3. Output

Informal definition of computation

Computation is a process that transforms strings into strings.
(We won't actually use this as a definition, but it could be done.)

Programs

We'd like to describe computations independently of input/output.
Program:

- Takes input
- Produces output
- Specifies computation as a **finite** sequence of steps to be applied to the input in order to produce the output. This specification does not depend on the input or output.

Program describes **many** different computations.

Modeling computation by programs

Arbitrary mappings between strings may be very complex.
Take an infinite set Σ^* (say, all strings).
The number of mappings from Σ^* to Σ^* is larger than the number of elements of Σ^*. (Diagonalization.)
On the other hand, each program has finite length and so they can be listed in a sequence.
Consequence: there are some computational problems that cannot be solved by a finite-length program. (Undecidability.)

Restrictions

Study computation as performed by specific computational devices (machines).
The processes studied at any moment include only those solvable by the machine.
We'll study three computation models in this course:

1. Finite Automata
2. Push-Down Automata
3. Turing Machines

Finite Automata

Input: a string over some given alphabet Σ.
Output: Yes/No.
Machine:

1. Reads input one symbol at a time.
2. At each moment, **is** in a unique **state**.
3. Before reading any input, is in **start state**.
4. If now is in state q and reads input letter a, there is a unique state $\delta(q, a)$ in which it will be the next moment.
5. Computation halts when input is exhausted.
6. If halted in an **accepting state**, output is Yes, otherwise output is No.

Finite Automaton Definition

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

- Q is a finite set (of *states*).
- Σ is a finite set (*alphabet*, its elements denoted as *letters* or *symbols*).
- δ is a function $\delta : Q \times Sigma \rightarrow Q$ (the *transition function*).
- q_0 is an element of Q (the *start state*).
- F is a subset of Q (the set of *accepting states*).

Finite Automata: once more

Input: a string over some given alphabet Σ.
Output: Yes/No.
Machine:

1. Reads input one symbol at a time.
2. At each moment, **is** in a unique **state**.
3. Before reading any input, is in **start state**.
4. If now is in state q and reads input letter a, there is a unique state $\delta(q, a)$ in which it will be the next moment.
5. Computation halts when input is exhausted.
6. If halted in an **accepting state**, output is Yes, otherwise output is No.

Acceptance by FA

A string u over Σ is *accepted* by a FA M, if after M reads u, starting from q_0, M ends up in an accepting state.
The set $L(M)$ of all strings accepted by M is called the **language of the machine M**. (We also say that M **accepts** or **recognizes** $L(M)$.)

Regular languages

Let Σ be an alphabet.
A set L of strings over Σ is **regular**, if there exists some FA M such that $L = L(M)$.
To study finite automata, it should be enough to study regular languages.

Some notation

String with no letters: ϵ.
If Σ is an alphabet, then Σ^* denotes the set of all strings over Σ.
Examples: $\Sigma = \{0\}$, $\Sigma = \{0, 1\}$, $\Sigma = \emptyset$.
A **language** is a set of strings: $L \subseteq \Sigma^*$.

Regular operations

Let $A, B \subseteq \Sigma^*$.
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: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$.
Theorem: The class of regular languages is closed under the regular operations.