Alphabets and Languages: the mathematics of strings

CSC320

Strings and symbols

- An *alphabet* is a finite set of *symbols*, e.g., the binary or Roman alphabet. We denote an arbitrary alphabet by Σ
- A *string* over an alphabet is a finite sequence of symbols from the alphabet.
- The *empty string* is the string with no symbols and is denoted ε.
- The *length* of a string is the number of symbols that it contains.
  - There is only one string of length 0. What is it?
- The length of a string w is denoted $|w|$.
- The symbol in the ith position is denoted w_i. We say that symbol w_i *occurs* in position i. A symbol may have more than one occurrence in a string.

Operations and relations on strings

- The operation of *concatenation* takes two string x and y and produces a new string xy by putting them together end to end. The string xy is called the *concatenation* of x and y.
  - Concatenation is an associative operation. So we will write, e.g., xyz for $(xy)z$ or $x(yz)$
- We write x^k for the string obtained by concatenating k copies of x.
- A string v is a *substring* of a string w iff there are strings x and y such that $w = xvy$. If $y = \varepsilon$ then v is a *suffix* of w. If $x = \varepsilon$ then v is a *prefix* of w.
- The *reversal* of a string w, denoted w^R is the string w "written backwards".

Languages: Sets of strings

- Σ^k is the set of all strings of length k from alphabet Σ.
- Σ^* is the set of all strings over alphabet Σ.
- Σ^+ is the set of all nonempty strings over alphabet Σ.
- A *language*, L, is set of strings over an alphabet.
  - That is, $L \subseteq \Sigma^*$
- We may apply set operations like *union*, *intersection*, and *set difference* to languages.
- The *complement* of a language L is $\Sigma^* - L$, and is denoted $\bar{L}$, if Σ is understood.

Kleene star

- If L_1 and L_2 are languages over Σ their *concatenation* is

$$L = L_1 L_2 = \{w \in \Sigma^* \mid w = xy \text{ for some } x \in L_1 \text{ and } y \in L_2\}$$

- The *Kleene star* of a language L, denoted L^*, is the set of all strings obtained by concatenating **zero** or more strings from L. Thus,

$$L^* = \{w = w_1 w_2 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in L\}$$

  - Examples: The star of Σ is Σ^*; The star of $\emptyset$ is $\{\varepsilon\}$

Languages, Specifications and Problems

- The vast majority of languages over a finite alphabet cannot be represented by a finite specification.
  - Why not?
- What languages can we specify? This is the primary question we will address in this course
- Recall from the first lecture that we said we will be concerned with *computational solutions* to *problems*.
- A problem is a mapping from *problem instances* to YES, NO.
- Languages may be viewed as an abstract representation of problems. For a problem Π, the associated language is

$$L_{\Pi} = \{x \in \Sigma^* : x \text{ is a } YES \text{ instance of } \Pi\}$$

- So studying "specifiable" languages is analogous to studying "solvable" problems.

Finite Automata

- The simplest computational model is the *finite automaton* (or *finite state machine*).
- Finite automata model systems with a *fixed* amount of memory, e.g, door controller.
  - Three components:
    - front pad, door, rear pad
    - door swings open to the rear
  - Door is in two possible *states*: CLOSED or OPEN
  - Four possible *signals* (or *inputs*): FRONT, REAR, BOTH, NEITHER
  - When CLOSED: NEITHER or BOTH → CLOSED, FRONT → OPEN; REAR → CLOSED
  - When OPEN: NEITHER → CLOSED; FRONT or REAR → OPEN; BOTH → OPEN

Formal Definition

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

- Q is a finite set of *states*,
- Σ is the input symbols or alphabet,
- $\delta: Q \times \Sigma \to Q$, the *transition function*, e.g., $\delta(q, a) = p$ where $q, p \in Q$ and $a \in \Sigma$
- $q_0 \in Q$ is the *start state*,
- $F \subseteq Q$ is the set of *accept states* or *final states*.
- The *language* of the machine M is the set L of all strings that M *accepts*. We write $L(M) = L$ and say M *accepts (or recognizes)* L. If the machine M accepts no strings then $L(M) = \emptyset$.

Example

- Consider the finite automaton $M_1 = (Q, \quad, \delta, q_1, F)$ where
  - $Q = \{q_1, q_2, q_3\}$, $\Sigma = \{0,1\}$, q_1 is the start state, $F = \{q_2\}$, and
  - δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

- We can also describe δ with a *state diagram* where states are represented by nodes and transitions are represented by direct edges labelled with the transition input symbols
  - The start state has an edge with no source state and the accept states have a double circle

Formal Definition of Computation (Acceptance)

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton, and let $w = w_1 w_2 \ldots w_n$ be a string over Σ.
- Then M *accepts* w if there is a sequence of states $r_0, \ldots, r_n$ in Q s.t.

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$
3. $r_n \in F$

- M *accepts* or *recognizes* L if $L = L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$.
- A language is called a *regular language* if some finite automaton accepts it.

More Examples

1. Consider $M_2 = (\{q_1, q_2\}, \{0,1\}, \delta, q_1, \{q_2\})$, where δ is

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

2. Construct the FA that accepts

$$L(M_3) = \{w \in \{a, b\}^* \mid w \text{ contains substring } ab\},$$

3. Construct the FA that accepts

$$L(M_4) = \{w \in \{0,1\}^* \mid w \text{ has even number of } 0's \text{ and } 1's\}$$

Closure Properties of Regular Languages

- A set is *closed* under some operation if applying that operation to elements of the set returns another element of the set.

Theorem: If L_1 and L_2 are regular languages, then so is $L_1 \cup L_2$.

Proof Idea: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be the automata that recognize L_1 and L_2, respectively. We construct $M = (Q, \Sigma, \delta, q_0, F)$ which recognizes $L_1 \cup L_2$ by *simulating* both of these machines *concurrently*, and accepting if one of them accepts.

Union Construction

1. Cartesian product of $Q = \{(r_1, r_2) \,|\, r_1 \in Q_1, r_2 \in Q_2\}$
2. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let
$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$
3. $q_0 = (q_1, q_2)$
4. $F = \{(r_1, r_2) | r_1 \in F_1 \, or \, r_2 \in F_2\}$

- Why this construction works?
  - remembers the pairs of states the machine is in ("concurrent execution".)
- Similarly, closed under intersection.

Closed Under Concatenation

Theorem: If L_1 and L_2 are regular languages, then so is $L_1 L_2$.

Proof Idea: Product construction doesn't help here… Need to do two consecutively, not concurrently. We need nondeterminism here!