Lecture Notes: Theory of Computation — Introduction (Course: MIT 18.404J)

Thobias K. Høivik

March 29, 2025

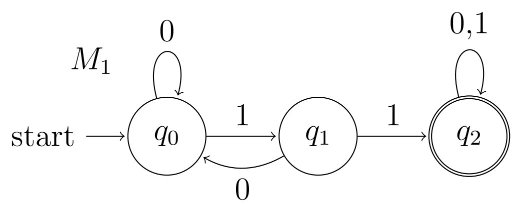
Computability Theory vs. Complexity Theory

Computability theory (prominent 1930s-1950s) is concerned with what is computable and not. **Complexity theory** (prominent 1960s-today) is concerned with what is computable in practice, and how hard is it to compute a particular problem. The first half of this course focuses on **Computability theory**. That means **Finite automata**, **Turing machines** and more.

The Role of Theory in Computer Science

1. Applications
2. Basic Research
3. Connections to other fields
4. What is the nature of computation?

Finite Automata



A **Finite Automaton** takes a finite string as input and outputs **accept** or **reject**. The computational process goes as follows:

1. Begin at start state
2. Read input symbols
3. Follow corresponding transitions

4. Accept if end with accept state, Reject if not

Consider the figure above. Then q_2 is an accepting state.
Examples:

$$01101 \rightarrow Accept$$

$$00101 \rightarrow Reject$$

After some thought we conclude that the strings w which the automaton M_1, seen above, accepts are in A where $A = \{w|w \text{ contains substring } 11\}$. We say that A is the language of M_1 and that M_1 recognizes A and that $A = L(M_1)$ (these are three equivalent statements).

Formal Definition

Definition 1. *A* ***finite automaton*** *M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$.*

- *Q finite set of states*
- *Σ finite set of alphabet symbols*
- *δ transition function $\delta : Q \times \Sigma \rightarrow Q$*
- *$q_0 \in Q$ starting state*
- *$F \subseteq Q$ set of accepting states*

Applying this definition to the automaton M_1 we can formalize as such:

$$\begin{gathered} M_1 = (Q, \Sigma, \delta, q_0, F) \\ Q = \{q_0, q_1, q_2\} \\ \Sigma = \{0, 1\} \\ F = \{q_3\} \\ \delta = \begin{array}{c|cc} \delta & 0 & 1 \\ \hline q_0 & q_0 & q_1 \\ q_1 & q_0 & q_2 \\ q_2 & q_2 & q_2 \end{array} \end{gathered}$$

String and Languages

- A **string** is a (usually) finite sequence of symbols in Σ.
- A **language** is a set of strings (finite or infinite).
- The **empty string** ϵ is the string of length 0.
- The **empty language** $\emptyset$ is the set with no strings (i.e the empty set)

Sidenote: $\{\epsilon\} \neq \emptyset$.

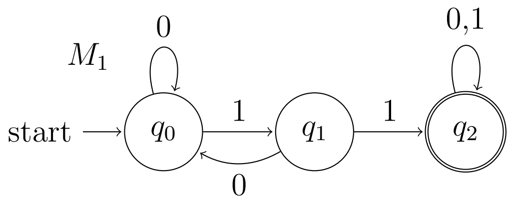
Definition 2. *M **accepts string** $w = w_1 w_2 \ldots 2_n, w_i \in \Sigma$ if there is a sequence of states $r_0, r_1, r_2, \ldots, r_n \in Q$ where:*

- $r_0 = q_0$
- $r_i = \delta(r_{i-1}, w_i)$ *for* $1 \leq i \leq n$
- $r_n \in F$

The language $L(M) = \{w | M \text{ accepts } w\}$ is the set of all strings accepted by M. We may also write $L(M)$ is the language of M or M recognizes $L(M)$.

Definition 3 (Regular Language)**.** *A language is regular if some finite automaton recognizes it.*

Recall M_1:



$L(M_1) = \{w | w \text{ contains substring } 11\} = A$. Therefore A is regular.

Regular Expressions

Let A, B be languages:
Union:

$$A \cup B = \{w | w \in A \vee w \in B\}$$

Concatenation:

$$A \circ B = \{xy | x \in A \wedge y \in B\} = AB$$

Star:

$$A^* = \{x_1 \ldots x_k | \forall x_i \in A, k \geq 0\}$$

A^* is like the powerset for a language so naturally $\epsilon \in A^*$ for all possible strings A. Another nice way to write it is

$$A^* = \bigcup_{n=0}^{\infty} A^n$$

where A^n is the strings of length n over the language A.
Regular expressions are built from Σ, members $\Sigma, \emptyset, \epsilon$ [Atomic], and by using $\cup, \circ, *$ [Composite].
Examples:

- $(0 \cup 1)^* = \Sigma^*$ gives all strings over Σ
- $\Sigma^*\{1\} = \Sigma^* 1$ gives all strings that end in 1.
- $\Sigma^* 11 \Sigma^*$ gives all strings that contain $11 = L(M_1)$

Closure Properties for Regular Languages

Theorem 1. *If A_1, A_2 are regular languages, so is $A_1 \cup A_2$ (closure under $\cup$).*

Proof. Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 and let $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2. To show $A_1 \cup A_2$ to be regular we must construct $M = (Q, \Sigma, \delta, q_0, F)$ recognizing $A_1 \cup A_2$. M should accept input w if either M_1 or M_2 accept w.
We construct M as follows:
- The states of M will be the union of the states of M_1 and M_2, plus a new initial state:

$$Q = Q_1 \cup Q_2 \cup \{q_0\}$$

where q_0 is the new initial state.
- The alphabet Σ remains the same as for M_1 and M_2.
- The transition function δ is defined as follows: - For each $q \in Q_1$, the transitions of M_1 are copied to M:

$$\delta(q, a) = \delta_1(q, a) \quad \text{for all } q \in Q_1, a \in \Sigma.$$

- Similarly, for each $q \in Q_2$, the transitions of M_2 are copied to M:

$$\delta(q, a) = \delta_2(q, a) \quad \text{for all } q \in Q_2, a \in \Sigma.$$

- Additionally, from the new initial state q_0, we add transitions to the initial states of M_1 and M_2 on ϵ-moves:

$$\delta(q_0, \epsilon) = \{q_1, q_2\}$$

- The initial state q_0 is the newly created initial state.
- The accepting states F of M are the union of the accepting states of M_1 and M_2:

$$F = F_1 \cup F_2$$

This ensures that M accepts a string if either M_1 or M_2 accepts it.
Since we have constructed a finite automaton M that recognizes $A_1 \cup A_2$, we conclude that $A_1 \cup A_2$ is a regular language.

□

Theorem 2. *If A_1, A_2 are regular languages, so is $A_1 A_2$ (closure under $\circ$)*

Proof. Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 and let $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2. To show that $A_1 A_2$ is regular, we need to construct a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ recognizing $A_1 A_2$, the concatenation of A_1 and A_2. We construct M as follows:
- States: The set of states Q in M will be the union of the states of M_1 and M_2, along with a new initial state q_0:

$$Q = Q_1 \cup Q_2 \cup \{q_0\}.$$

- Alphabet: The alphabet Σ remains the same as for M_1 and M_2.
- Transition functionδ: - The transition function for states in M_1 and M_2 will be copied directly from the respective machines:

$$\delta(q, a) = \delta_1(q, a) \quad \text{for all } q \in Q_1, a \in \Sigma$$

and

$$\delta(q, a) = \delta_2(q, a) \quad \text{for all } q \in Q_2, a \in \Sigma.$$

- Additionally, we modify the transitions of M to connect the final states of M_1 to the initial states of M_2:

$$\delta(f_1, \epsilon) = q_2 \quad \text{for each } f_1 \in F_1.$$

This ensures that after reaching an accepting state of M_1, the automaton can move to the initial state of M_2 without consuming any additional input (via an ϵ-transition).
- Initial state: The initial state of M will be the new state q_0, which transitions to the initial state q_1 of M_1 via an ϵ-transition:

$$\delta(q_0, \epsilon) = q_1.$$

- Accepting states: The accepting states of M will be the accepting states of M_2, because the machine will accept if it reaches an accepting state in M_2 after processing a string in A_1 followed by a string in A_2:

$$F = F_2.$$

Since we have constructed a finite automaton M that recognizes $A_1 A_2$, we conclude that the concatenation of A_1 and A_2 is a regular language.

□