Introduction to Computer Science Theory

Alvin Lin

August 2017 - December 2017

Abstract Machines

Our goal is to answer fundamental computer science questions:

- What can and cannot be computed?
- What can and cannot be computer efficiently?

To answer this, we need a **universal model of computation**. We also need **restricted models of computation**, such as string matching, lexical analysis, and parsing.

Finite Automata

This is our most restricted model, and it can be represented as input on tape that reads input once from left to right. It has no memory except for one register.

```
[a a b a c a d a b a d a]
```

They have finite control and contain a state. We call these **deterministic finite automata**.

What languages can finite automatons recognize?

Given a language, we want to write a program that given a string, can tell whether or not the string is in the language. For this example, we will define our language as the strings over $\{a, b\}$ that contain an even number of a's.

$$\{aab, aba, aa, b, \epsilon\} \in L$$

$$\{ab, a, abb\} \notin L$$

One algorithm:

- Let $n_a \leftarrow 0$
- For $i \in \{1, \ldots, n\}$: If $x_i == a$, let $n_a \leftarrow n_a + 1$.
- If n_a is even, accept.
- Else, reject.

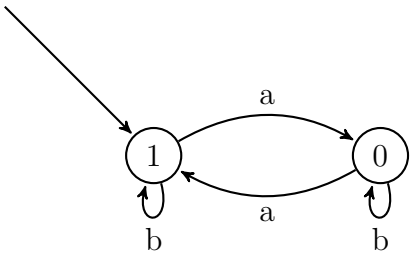
This algorithm cannot be run on a finite automaton because the n_a counter can get arbitrarily large. A finite automata can only have a finite number of states.

Example

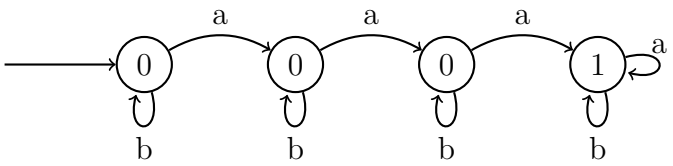
Give **set builder notation** and a **transition diagram** for a finite automaton that recognizes the language that consists of the strings over $\{a, b\}$ that:

- contain an even number of a's.

$$\{x \in \{a, b\}^* \mid x \; has \; 0 \; mod \; 2 \; a's\}$$



- contain at least 3 a's.



Formally defined, a finite automata is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set of *states*
- Σ is an alphabet of *input symbols*
- δ is a total function from $Q \times \Sigma$ to Q: the *transition function* (δ is often given as a *transition table*)
- $q_0 \in Q$ is the *initial state*
- $F \subseteq Q$ is the set of *accepting* states

For the example above of a finite automaton that contains an even number of a's: $M = (Q, \Sigma, \delta, 0, \{0\})$, where:

- $Q = \{0, 1\}$
- $\Sigma = \{a, b\}$
- δ: $Q \times \Sigma \to Q$ is defined on $(q, z) \in Q \times \Sigma$ as:

$$\delta(q, x) = \begin{cases} q & if\ x = b \\ q + 1 \mod 2 & otherwise \end{cases}$$

Acceptance on Deterministic Finite Automata

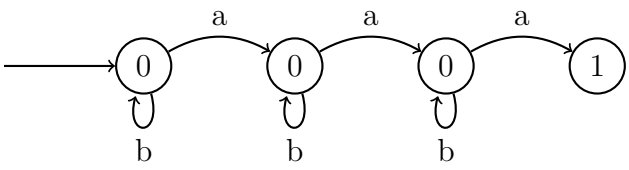
For finite automaton $M = (Q, \Sigma, \delta, q_\circ, F)$, $\delta(q, a)$ is the state M goes to if it is in state q and reads an a. We say that M accepts $w = w_1 w_2 \dots w_n \in \Sigma^*$ if there is a sequence (repeats allowed) of states $r_0, r_1, \dots, r_n \in Q$ such that:

1. $r_0 = q_0$
2. $\forall i \in \{0, \dots, n-1\}(\delta(r_i, w_{i+1}) = r_{i+1})$
3. $r_n \in F$

A deterministic finite automaton M recognizes a language A if $A = \{w \mid M\ accepts\ w\}$. We say $A = L(M)$. A language is called a **regular language** if some finite automaton recognizes it.

Nondeterminism

Look at the following transition diagram for the set of strings over $\{a, b\}$ that contain at most 2 a's.



This is not the transition diagram of a finite automaton because the last state is missing transitions. Nondeterminism removes the restrictions on finite automata and allows for a transition diagram to omit transitions on certain inputs or multiple transitions on certain inputs. Formally defined, a nondeterministic finite automaton is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$ where:

- Q is a finite set of *states*
- Σ is an alphabet of *input symbols*
- δ is a total function from $Q \times \Sigma_\epsilon$ to 2^Q
- q_0 is the initial state
- $F \subseteq Q$ is the set of *accepting states*

Acceptance on Nondeterministic Finite Automata

For finite automaton $M = (Q, \Sigma, \delta, q_0, F)$, $\delta(q, a)$ is the state M goes to if it is in state q and reads an input a. We say that M accepts $w = w_1 w_2 \dots w_n$ where each $w_i \in \Sigma_\epsilon$ if there is a sequence of states $r_0, r_1, \dots, r_n \in Q$ such that:

- $r_0 = q_0$
- $\forall i \in \{0, \dots, n-1\}(r_{i+1} \in \delta(r_i, w_{i+1}))$
- $r_n \in F$

You can find all my notes at `http://omgimanerd.tech/notes`. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech