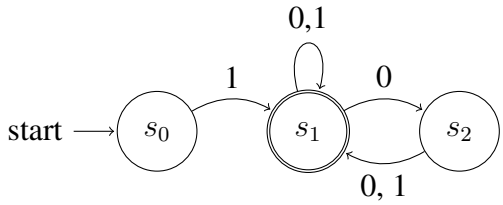
Computation Theory Note

Ran Xie

March 19, 2023

1 Regular Languages



state diagram of **finite automaton** M. q_0, q_1, q_2 are **states**. q_0 is **start state**. q_1 is **accept state**. Arrows are called **transitions**. Input is string of 0s and 1s. The output is either **reject** or **accept**.

(Deterministic) Finite Automaton (DFA) M is 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

1. Q is a finite set called the states
2. Σ is a finite set called the alphabet
3. $\delta : Q \times \Sigma \to Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states.

Language of machine M, A is the set of all strings that M accepts and write $L(M) = A$. We say M accepts A or M recognizes A.

Language operations: Let A and B be languages

1. Union: $A \cup B = \{x | x \in A \text{ or } x \in B\}$
2. Concatenation: $A \circ B = \{xy | x \in A, y \in B\}$
3. Star: $A^* = \{x_1 x_2 \dots x_k | k \geq 0 \text{ and } x_i \in A\}$

Regular Language is a language accepted by some finite automatons. The class of regular language is closed under union, concatenation and star operation.

Nondeterministic finite automaton (NFA) is 5-tuple $(Q, \Sigma, \delta, q_0, F)$.

1. Q is finite set of states
2. Σ is finite alphabet
3. $\delta : Q \times \Sigma_\epsilon \longrightarrow P(Q)$
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states

Remark: ϵ is empty string. The transition function maps current input or ϵ and the current state to produce a set of next states (in the power set $P(Q)$) rather than a single next state in DFA. NFA accepts $w = y_1 y_2 \dots y_m$ if $y_i \in \Sigma_\epsilon$ for all i and a sequence of states $r_0, \dots, r_m \in Q$ exists such that

- $r_0 = q_0$
- $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0, \dots, m-1$
- $r_m \in F$

This means even NFA branches out but if one sequence of states exists such that the last state is in F, the NFA accepts the input.

Result 1 *Two automatons are equivalent if they recognizes the same language. Every NFA has an equivalent DFA. It follows that a language is regular if some NFA recognizes it.*

R is **Regular Expression** if R is

1. a for some a in the alphabet Σ
2. ϵ
3. $\emptyset$
4. $(R_1 \cup R_2)$ (Union)
5. $(R_1 \circ R_2)$ (Concatenation)
6. (R_1^*) (Star)

where R_1 and R_2 are regular expressions. This type of definition is called **Inductive definition** (not a circular one).

Result 2 *A language is regular iff some regular expression describes it.*

A **Generalized nondeterministic finite automaton (GNFA)** is a 5-tuple $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$ where

1. Q is the finite set of states
2. Σ is the input alphabet
3. $\delta : (Q - \{q_{\text{start}}\}) \times (Q - \text{accept}) \longrightarrow R$ is the transition function.
4. q_{start} is the start state and q_{accept} is the accept state.

2 Context Free Grammar

Finite automata and regular expressions are methods to describe languages. Context-Free grammar is a more powerful method of describing languages. Languages described as called Context Free Languages.

A **grammar** consists of a collection of substitution rules also called **productions**. E.g.:

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

$\{A, B\}$ are variable symbols. $\{0, 1, \#\}$ are terminal symbols. A is the start variable. We say A yields $0A1$ written $A \Rightarrow 0A1$. A derives $0\#1$ written $A \overset{*}{\Rightarrow} 0\#1$ because there exists $A \Rightarrow 0A1 \Rightarrow 0B1 \Rightarrow 0\#1$

Formally, A **Context Free Grammar** is a 4-tuple (V, Σ, R, S) where

1. V is a finite set called the variables
2. Σ is a finite set, disjoint from V called the terminals
3. R is a finite set of rules, with each rule being a variable and a string of variables and terminals
4. $S \in V$ is the start variable.

Example: $G = (\{S\}, \{a, b\}, R, S)$ The set of rules R is $S \to aSb|SS|\epsilon$

Example: $G = (V, \Sigma, R, \langle\text{EXPR}\rangle)$ where $V = \{\langle\text{EXPR}\rangle, \langle\text{TERM}\rangle, \langle\text{FACTOR}\rangle\}$ and $\Sigma = \{a, +, \times, (,)\}$

$$\langle\text{EXPR}\rangle \to \langle\text{EXPR}\rangle + \langle\text{TERM}\rangle|\langle\text{TERM}\rangle$$
$$\langle\text{TERM}\rangle \to \langle\text{TERM}\rangle \times \langle\text{FACTOR}\rangle|\langle\text{FACTOR}\rangle$$
$$\langle\text{FACTOR}\rangle \to (\langle\text{EXPR}\rangle)|a$$

The rule produces $a + a \times a$, $(a + a) \times a$.

A string w is a **leftmost derivation** if every derivation step the leftmost variable is the one that gets substituted. If a context-free grammar G has two or more different leftmost derivation, then it is **ambiguous**. (Some CFGs are inherently ambiguous)

A context-free grammar is in **Chomsky normal form** if every rule is of the form

$$A \to BC$$
$$A \to a$$

where a is terminal and A, B, C are variable (B, C cannot be start variables).

Result 3 *Any context-free language is generated by a context-free grammar in* ***Chomsky normal form***.