
Introduction to the Theory of Computation (Chapter 1 – 2)

Jiakang Lu
Qual Study Group
June 28th, 2007

Overview

(Part One: Automata and Languages)

- Regular Languages
 - DFA
 - NFA
 - Regular Expressions
 - Pumping Lemma v1
- Context-Free Languages
 - Context-Free Grammars
 - PDA
 - Pumping Lemma v2

Deterministic Finite Automata

- **Definition:** DFA $M = (Q, \Sigma, \delta, s, F)$, where
 - Q is finite set of states
 - Σ is input alphabet
 - $\delta: Q \times \Sigma \rightarrow Q$ transition function
 - $s \in Q$ is initial state
 - $F \subseteq Q$ is set of final states

The Language of DFA

- $L(M)$ - Language accepted by M
- Define extended transition function δ^* :
 - $\delta^*(q, \varepsilon) = q$
 - $\delta^*(q, w\sigma) = \delta(\delta^*(q, w), \sigma)$
- **Definition:** $L(M) = \{ w \text{ in } \Sigma^* \mid \delta^*(s, w) \text{ in } F \}$
 - the set of string w that take the start state s to one of the accepting states

Regular Languages

- **Definition:** A language is **regular** iff there is a DFA that accepts it.
- *Closure Properties* of Regular Languages:
 - Regular operations:
 - union, concatenation, star
 - Also other operations:
 - intersection, complement, difference
 - reversal
 - homomorphism, inverse homomorphism

Non-deterministic Finite Automata

- **Definition:** NFA $N = (Q, \Sigma, \delta, s, F)$ where:
 - Q - finite set of states
 - Σ - input alphabet
 - δ is a subset of $Q \times \Sigma \rightarrow P(Q)$, transition function
 - s - initial state
 - $F \subseteq Q$ - set of final states
- The only difference between DFA and NFA is in the type of value that δ returns

The Language of NFA

- Definition:

$L(N) = \{ w \text{ in } \Sigma^* \mid \delta^*(s,w) \cap F \text{ is nonempty} \}$

- the set of strings w in Σ^* such that $\delta^*(s,w)$ contains at least one accepting state

NFA vs. DFA

- Is **NFA** more powerful than **DFA**?
 - **Ans:** No.
- **Theorem:**
 - For every NFA M there is an equivalent DFA M'
- **Proof Idea:**
 - NFA is in a set of states at any point during reading a string.
 - DFA will use a lot of states to keep track of this.
- **Important Assumption:**
 - No transition labeled by epsilon.
 - (Will get rid of this assumption later.)

ε -Transitions

- Define ε -closure of state q as $\delta^*(q, \varepsilon)$.
 - ε -closure(q) = {all states reachable from q using only ε -transitions}
- Extend ε -closure to sets of states by:
 - ε -closure($\{s_1, \dots, s_m\}$) =
 ε -closure(s_1) \cup ... \cup ε -closure(s_m)

Formal Definition of e-NFA

- e-NFA $N = (Q, \Sigma, \delta, s, F)$ where:
 - Q - finite set of states
 - Σ - input alphabet
 - δ is a subset of $Q \times \Sigma_\epsilon \rightarrow P(Q)$, transition function
 - s - initial state
 - $F \subseteq Q$ - set of final states
- Notation: $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$.

Regular expressions

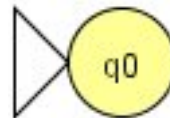
- A **third** way to view regular languages.
- A language **generator** model instead of language acceptor.
- **Definition:**
 - The smallest class of strings over $\Sigma \cup \{ (,), *, \cup, \bullet, \phi \}$ that includes:
 1. ϕ (basic reg. exp.)
 2. σ for every σ in Σ . (basic reg. exp.)and is **closed** under \cup , $*$ and \bullet

Regular Expressions vs. FA's

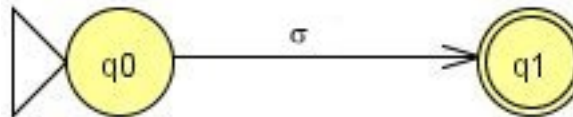
- Regular expressions **generate** exactly the class of regular languages.
- Theorem:
 - (a) For every regular expression, there is an equivalent ε -NFA
 - (b) For every DFA, there is an equivalent regular expression.

- Proof of (a):

- For ϕ , the NFA is:



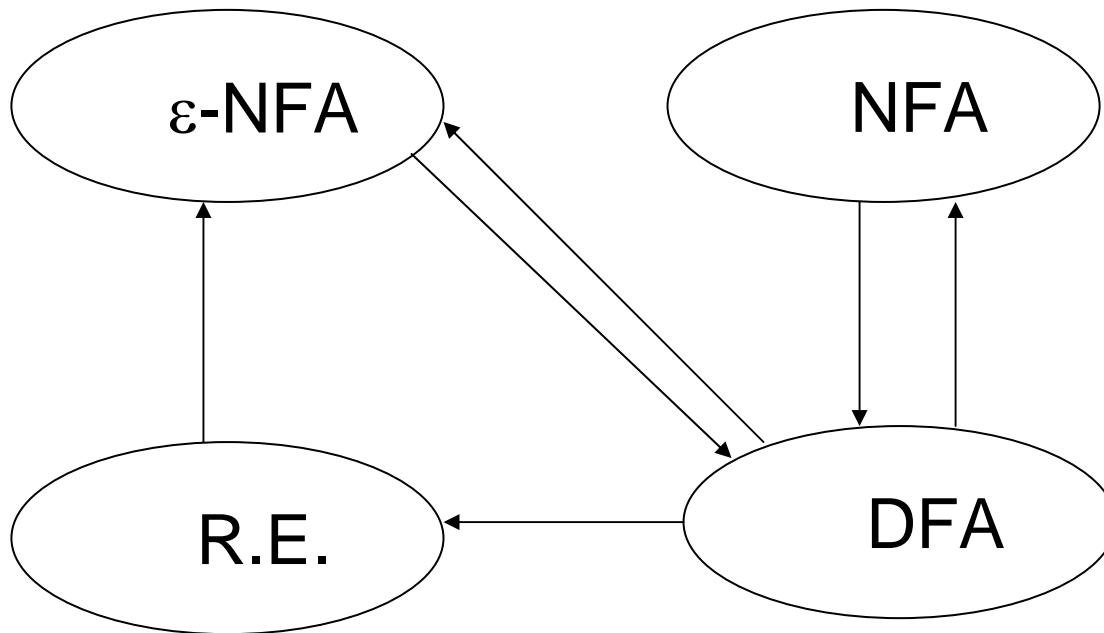
- For σ , the NFA is:



- For composite regular expressions: use closure under \cup , \bullet and $*$

Summary of Equivalence Relationship

- Four different notations for regular languages



Pumping Lemma v1

- If A is a regular language, then there is a number p (the pumping length) where, if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:
 1. for each $i \geq 0$, $xy^iz \in A$,
 2. $|y| > 0$, and
 3. $|xy| \leq p$.

Using Pumping Lemma

- The typical application of pumping lemma, to show that a language is **not** regular.
- You **must** choose string w so that $w \in L$ and $|w|$ is at least the pumping length.
 - **Example:** choosing $w = aaabbb$ is **wrong** since we do not know the exact value of p .
- You **must** consider all possibilities for x , y and z such that $w = xyz$ and $|xy| \leq p$.
- The pumping lemma **CANNOT** be used to show that a language is regular, since it assumes that L is regular.

Some Tips

- Closure properties can be used effectively for:
 - shortening cumbersome Pumping lemma arguments
 - **Example:** $\{w \text{ in } \{a, b\}^* \mid w \text{ has equal a's and b's}\}$
 - showing that certain languages are regular
 - **Example:** $\{w \text{ in } \{a, b\}^* \mid w \text{ begins with a and } w \text{ contains a b}\}$.

Context-Free Grammar

- **Definition:** CFG $G = (V, \Sigma, R, S)$, there
 - V is finite set of variables
 - Σ is finite set, disjoint from V , of terminals
 - R is finite set of rules
 - $S \in V$ is the start variable

Derivations and $L(G)$

- One step derivation:
 - $u \Rightarrow v$ if $u = xAy$, $v = xwy$ and $A \rightarrow w$ in P
- Zero or more steps derivation:
 - $u \Rightarrow^* v$ if $u = u_0 \Rightarrow u_1 \Rightarrow \dots \Rightarrow u_n = v$ ($n \geq 0$)
- $L(G) = \{ w \text{ in } T^* \mid S \Rightarrow^* w \}$.
- A language L is **context-free** if there is a CFG G with $L(G) = L$

Ambiguity

- A CFG is **ambiguous** if there is a string with at least two leftmost derivations

- Example:

$E \rightarrow E + E \mid E * E \mid (E) \mid x \mid y$ is ambiguous

- A CFL is **inherently ambiguous** if every CFG that generates it is ambiguous.

- Example:

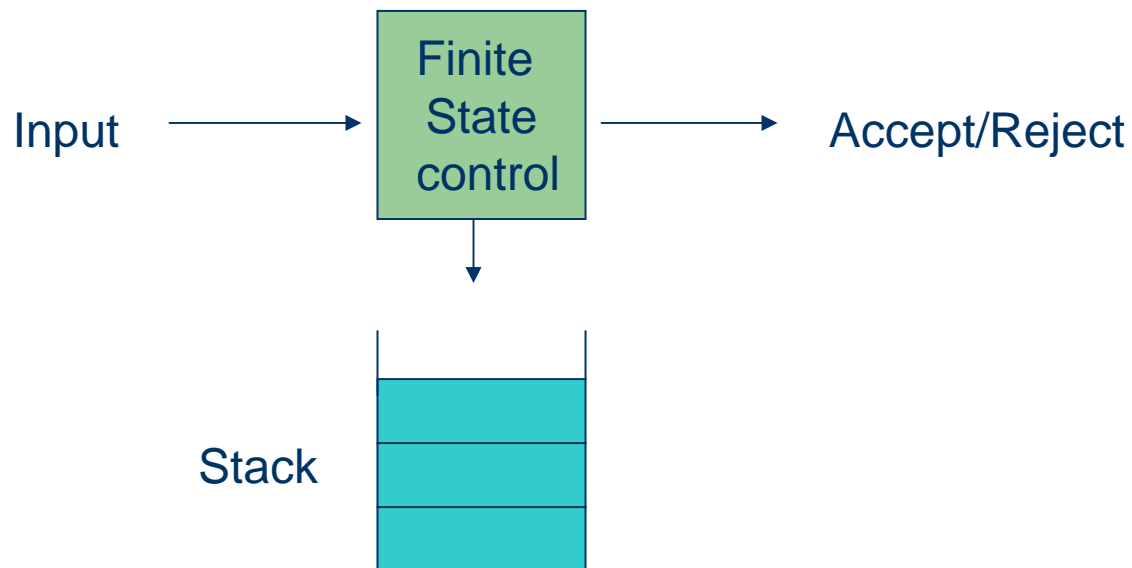
$\{a^n b^n c^m \mid n, m \geq 0\} \cup \{a^m b^n c^n \mid n, m \geq 0\}$

Chomsky Normal Form (CNF)

- Rules of CFG G are in one of **two** forms:
 - (i) $A \rightarrow a$
 - (ii) $A \rightarrow BC$, $B \neq S$ and $C \neq S$
- + Only **one** rule of the form $S \rightarrow \varepsilon$ is allowed if ε in $L(G)$.
- Benefits:
 - Easier to reason with in proofs
 - Leads to more efficient algorithms

Push Down Automata (PDA)

- Language **Acceptor** Model for CFLs
- It is an **NFA** with a stack.



Formal Definition of PDA

- **Definition:** PDA $M = (Q, \Sigma, \Gamma, \delta, s, F)$, where
 - Q is finite set of states
 - Σ is the input alphabet
 - Γ is the stack alphabet
 - $\delta \subseteq (Q \times \Sigma_\epsilon \times \Gamma_\epsilon) \rightarrow P(Q \times \Gamma_\epsilon)$
 - $s \in Q$ is the start state
 - $F \subseteq Q$ is the set of final states

The Language of PDA

- **Define** extended transition function δ^* as:

$$(1) \delta^*(q, \varepsilon, \varepsilon) = \{(q, \varepsilon, \varepsilon)\} \cup \{(p, \varepsilon, \varepsilon) \mid ((q, \varepsilon, \varepsilon), (p, \varepsilon)) \in \delta\}$$

$$(2) \delta^*(q, uv, xy) = \bigcup \{\delta^*(p, v, wy) \mid ((q, u, x), (p, w)) \in \delta\}$$

- M accepts w

- if (f, ε, x) in $\delta^*(s, w, \varepsilon)$, where x in Γ^*

- if $(f, \varepsilon, \varepsilon)$ in $\delta^*(s, w, \varepsilon)$ [**we use**]

- $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$

- “final state language”

- “empty stack language” [**we use**]

Context-Free Languages

- **Definition:** A Language is **context-free** iff there is a PDA that accepts it.
- *Closure Properties* of Context-Free Languages:
 - Regular operations:
 - union, concatenation, star
 - Also other operations:
 - reversal
 - homomorphism, inverse homomorphism
 - But not under
 - intersection, complement, difference
- CFLs are closed under intersection with regular languages.

PDA's vs. CFG's

■ Theorem:

- For every CFG G there is a PDA M such that $L(G) = L(M)$
- For every PDA M there is a CFG G such that $L(M) = L(G)$

Pumping Lemma v2

- If A is an infinite context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p , then s may be divided into five pieces $s = uvxyz$ satisfying the conditions.
 1. For each $i \geq 0$, $uv^ixy^iz \in A$,
 2. $|vy| > 0$, and
 3. $|vxy| \leq p$.

Using Pumping Lemma

- When s is divided into $uvxyz$, cond. 2 says - either v or y is not the empty string.
 - Otherwise the theorem would be trivially true.
- Cond. 3 say - the pieces v , x , and y together have length at most p .
 - This condition is useful in proving that certain languages are not context free.

Some Tips

- Closure properties can be used effectively for:
 - shortening cumbersome Pumping lemma arguments
 - **Example:** $\{w \text{ in } \{a, b, c\}^* \mid w \text{ has equal } a\text{'s, } b\text{'s, and } c\text{'s}\}$.
 - showing that certain languages are context-free
 - **Example:**
 $\{w \text{ in } \{a, b, c\}^* \mid w \text{ has equal } a\text{'s and } b\text{'s or equal } b\text{'s and } c\text{'s}\}$.