Automata

DFA
A Deterministic Finite Automata is defined as $M = (Q, \Sigma, \delta, q_0, F)$ where
- Q: A *finite* set of states
- Σ: A finite set of symbols
- δ: Transition function
  - $\delta : Q \times \Sigma \to Q$
- F: A set of accept states
  - $F \subset Q$

Notes:
- $F = \emptyset \Rightarrow L(M) = \emptyset$

NFA
A Nondeterministic Finite Automata is defined as $M = (Q, \Sigma, \delta, q_0, F)$ where
- Q: A *finite* set of states
- Σ: A finite set of symbols
  - Let $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$
- δ: Transition function
  - $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$
- F: A set of accept states

Notes:
- Implicit reject when no transition is labeled

PDA
A Pushdown Automata is defined as $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ where
- Q: Set of states
- Σ: Input alphabet
- Γ: Stack alphabet
- δ: $Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \Rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$
- q_0: Start state
- F: Set of accept states

CFG
A CFG is defined as (V, Σ, R, S) where
- V: finite set of variables
- Σ: finite set of terminals
- R: finite set of rules
- S: start variable

Closure

Regular Languages
Closed under:
- Union
- Concatenation
- Star
- Intersection
- Complement

CFLs
Closed under:
- Union
- Concatenation
- Star

If A_1 is a regular language and A_2 is a CFL, then $A_1 \cup A_2$ is a CFL

Pumping Lemmas

Regular Languages
$\forall$ Regular Language $A, \exists\, p \in \mathbb{Z}^{>0}$ s.t. $s \in A \wedge |s| \geq p \Rightarrow s$ may be divided into 5 pieces, $s = xyz$ s.t.

1) $xy^iz \in A \;\forall\; i \geq 0$
2) $|y| > 0$
3) $|xy| \leq p$

CFLs
$\forall$ CFL $A, \exists\, p \in \mathbb{Z}^{>0}$ s.t. $s \in A \wedge |s| \geq p \Rightarrow s$ may be divided into 5 pieces, $s = uvxyz$ s.t.

1) $uv^ixy^iz \in A \;\forall\; i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Other

CNF
A CFG is in Chomsky Normal Form if every rule is of the form:

$$A \to BC$$

$$A \to a$$

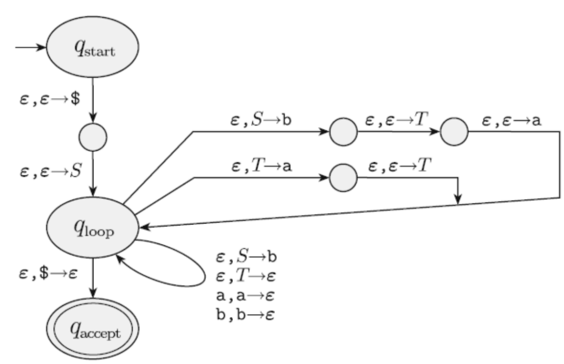
and
- a is a terminal
- A, B, C are variables
- Start variable cannot be on the right side of any rule
- $S \to \varepsilon$ is permitted **only** for the start variable S

Converting to CNF
1. Add a new start variable S_0 and rule $S_0 \to S$, where S is the original start variable
2. Eliminate all ε-rules of the form $A \to \varepsilon$, where $A \neq S_0$
   - For each occurrence of A on the right side of a rule, add a new rule with that occurrence removed
3. Eliminate unit rules of the form $A \to B$
   - For any rule $B \to u$, add $A \to u$
   - Including the start variable
4. Convert remaining rules to proper form
   - For each rule $A \to u_1u_2...u_k, k \geq 3$, where each u_i is a variable or terminal:
     - Remove the rule and add the following rules:
       - $A \to u_1A_1$,
       - $A_1 \to u_2A_2$,
       - $A_{k-2} \to u_{k-1}A_k$,

Converting CFG to PDA
We use the procedure developed in Lemma 2.21 to construct a PDA P_1 from the following CFG G.

$$S \to \mathtt{a}T\mathtt{b} \mid \mathtt{b}$$
$$T \to T\mathtt{a} \mid \varepsilon$$

The transition function is shown in the following diagram.



1. Place "\$" and then the start variable on the stack
2. Repeat the following steps:
   a. If the top of the stack is a variable A, then:
      - Nondeterministically select one of the rules for A and substitue A with the right side of the rule
   b. If the top of the stack is a terminal a, then:
      - Read the next symbol from the input and compare it with a. If not match, reject the branch
   c. If the top of the stack is \$, then:
      - If all input has been read, accept.