# CS3231 Midterm 2 Printed Notes

Chong Chin Herng

October 2024

## 6 CFG

**Context-Free Grammar (CFG)** A CFG $(V, T, P, S)$ is a method to describe a language over $\Sigma$ where

- $V$ is a finite set of non-terminal symbols
- $T$ is a finite set of terminal symbols such that $V \cap T \neq \emptyset$
- $P$ is a finite set of productions of the form $A \to \gamma$, where $A \in V$ and $\gamma \in (V \cup T)^*$
- $S \in V$ is the start symbol

Additionally, for any $\alpha, \beta, \gamma \in (V \cup T)^*$, if there is a production $A \to \gamma$ in $P$, then $\alpha A \beta$ derives $A$ in one step, or $\alpha A \beta \Rightarrow \alpha \gamma \beta$. Moreover, $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$. Finally, $L(G) = \{w \in T^* | S \Rightarrow^* w\}$.

**Context-Free Language (CFL)** A CFL is a language that can be described by a CFG.

**Derivation** A derivation of a string $w$ is a sequence of derivation steps from the start symbol to $w$.

**Sentential Form** $\alpha \in (V \cup T)^*$ is a sentential form with respect to a CFG with start symbol $S$ if and only if $S \Rightarrow^* \alpha$.

**Left Most Derivation** A left most derivation is a derivation such that in each step, the leftmost non-terminal in the sentential form is replaced.

**Right Most Derivation** A right most derivation is a derivation such that in each step, the rightmost non-terminal in the sentential form is replaced.

**Parse Tree** A parse tree is a graphical representation of a derivation.

**Right-Linear Grammar** A CFG $(V, T, P, S)$ is right-linear if and only if all its productions are of the form $A \to wB$ or $A \to w$ for some $A, B \in V$ and $w \in T^*$.

**Theorem 6.1** There is a right-linear grammar that can describe a regular language.

**Theorem 6.2** Languages described by a right-linear grammar are regular.

**Ambiguous Grammar** A CFG $G$ is ambiguous if and only if there exists $w \in L(G)$ such that there exists two different parse trees for the derivation of $w$.

**Inherently Ambiguous Language** A language $L$ is inherently ambiguous if and only if any CFG that describes $L$ is ambiguous.

# 7 Push Down Automata (NPDA/PDA)

**Push Down Automaton (PDA)** A PDA $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a finite automaton where

- $Q$ is a finite set of states
- $\Sigma$ is a finite set of input symbols
- $\Gamma$ is a finite set of stack symbols
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ is a transition function
- $q_0 \in Q$ is the starting state
- $Z_0 \in \Gamma$ is the initial stack symbol
- $F \subseteq Q$ is a finite set of accepting states

**Instantaneous Description of a PDA** An instantaneous description of a PDA with set of states $Q$, set of input symbols $\Sigma$ and set of stack symbol $\Gamma$ is a tuple $(q, w, \alpha)$ where $q \in Q$, $w \in \Sigma^*$ and $\alpha \in \Gamma^*$.

**Step of a PDA** For any PDA $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, if $(p, \beta) \in \delta(q, a, X)$, then for any $w \in \Sigma^*$ and $\alpha \in \Gamma^*$, the instantaneous description $(q, aw, X\alpha)$ yields the instantaneous description $(p, w, \beta\alpha)$ in one step, or $(q, aw, X\alpha) \vdash (p, w, \beta\alpha)$. Moreover, $\vdash^*$ is the reflexive and transitive closure of $\vdash$.

**Language accepted by a PDA by final state** A language $L$ is accepted by a PDA $P$ with input symbol set $\Sigma$ by final state if and only if $L = \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q_f, \epsilon, \alpha) \text{ for some } q_f \in F\}$.

**Language accepted by a PDA by empty stack** A language $L$ is accepted by a PDA $P$ with input symbol set $\Sigma$ by empty stack if and only if $L = \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon) \text{ for some } q \in Q\}$.

**Theorem 7.1** There is a PDA that accepts a given language by final state if and only if there is a PDA that accepts the same language by empty stack.

**Theorem 7.2** There is a PDA that accepts a given language if and only if there is a CFG that describes the same language.

**Deterministic Push Down Automaton (DPDA)** An DPDA $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a finite automaton where

- $Q$ is a finite set of states
- $\Sigma$ is a finite set of input symbols
- $\Gamma$ is a finite set of stack symbols
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ is a transition function
- $q_0 \in Q$ is the starting state
- $Z_0 \in \Gamma$ is the initial stack symbol
- $F \subseteq Q$ is a finite set of accepting states

such that for all $a \in \Sigma \cup \{\epsilon\}, Z \in \Gamma$ and $q \in Q$,

- $|\delta(q, a, Z)| \leq 1$
- If $\delta(q, \epsilon, Z) \neq \emptyset$, then if $a \neq \epsilon$, then $\delta(q, a, Z) = \emptyset$

Instantaneous descriptions and steps of DPDA is similar to that of NPDA.

**Theorem 7.3** There exists a language accepted by a PDA but not by any DPDA.

**Theorem 7.4** There is a DPDA accepting by final state that can accept a regular language.

**Theorem 7.5** If $L$ is accepted by a DPDA by empty stack, then for every $x, y \in L$, $x$ is not a prefix of $y$.

# 8 CFG Properties

**Chomsky Normal Form** A CFG $(V, T, P, S)$ is in Chomsky Normal Form if and only if each of its productions are of the form $A \to BC$ or $A \to a$ for some $A, B, C \in V$, $a \in T$.

**Useful Symbol** For a CFG $(V, T, P, S)$, a symbol $A$ is useful if and only if $S \Rightarrow^* \alpha A \beta \Rightarrow^* w$ for some $\alpha, \beta \in (V \cup T)^*$ and $w \in T^*$.

**Generating Symbol** For a CFG $(V, T, P, S)$, a symbol $A$ is generating if and only if $A \Rightarrow^* w$ for some $w \in T^*$.

**Reachable Symbol** For a CFG $(V, T, P, S)$, a symbol $A$ is reachable if and only if $S \Rightarrow^* \alpha A \beta$ for some $\alpha, \beta \in (V \cup T)^*$.

**Theorem 8.1** If a symbol is useful, then it is reachable and generating.

**Algorithm 8.2** Given a CFG $G = (V, T, P, S)$, the algorithm below removes all non-generating symbols from $G$.

1. All symbols in $T$ are generating.
2. If there is a production of the form $A \to \alpha$ in $P$ and $\alpha$ consists only of generating symbols, then $A$ is generating.
3. Repeat step 2 until no new generating symbols are discovered.
4. The remaining symbols are non-generating. Remove all productions involving them.

**Algorithm 8.3** Given a CFG $G = (V, T, P, S)$, the algorithm below removes all non-reachable symbols from $G$.

1. $S$ is reachable.
2. If $A$ is reachable and there is a production of the form $A \to \alpha$ in $P$, then every symbol in $\alpha$ is reachable.
3. Repeat step 2 until no new reachable symbols are discovered.
4. The remaining symbols are non-reachable. Remove all productions involving them.

**Algorithm 8.4** Given a CFG $G$, the algorithm below removes all useless symbol from $G$.

1. Remove non-generating symbols from $G$.
2. Remove non-reachable symbols from $G$.

$\epsilon$ **Productions** An $\epsilon$ production is a production of the form $A \to \epsilon$.

**Nullable Symbol** For a CFG $(V, T, P, S)$, if there is a production of the form $A \to \epsilon$ in $P$, then $A$ is nullable. If there is a production of the form $A \to \alpha$ in $P$ and every symbol in $\alpha$ is nullable, then $A$ is nullable.

**Algorithm 8.5** Given a CFG $G = (V, T, P, S)$ such that $\epsilon \notin L(G)$, the algorithm below removes all $\epsilon$ productions from $G$.

1. Identify nullable symbols.
2. Remove $\epsilon$ productions.
3. For each production of the form $B \to \alpha$ in $P$, replace it with all possible productions of the form $B \to \alpha'$ where $\alpha'$ can be formed from $\alpha$ by deleting zero or more nullable symbols.

**Unit Production** For a CFG $(V, T, P, S)$, a unit production is a production of the form $A \to B$ for some $A, B \in V$.

**Unit Pair** For a CFG $(V, T, P, S)$, for any $A, B, C \in V$, $(A, A)$ is a unit pair. If $(A, B)$ is a unit pair and there is a production of the form $B \to C$ in $P$, then $(A, C)$ is a unit pair.

**Algorithm 8.6** Given a CFG $G = (V, T, P, S)$, the algorithm below removes all unit productions from $G$.

1. Identify unit pairs.
2. Remove unit productions.
3. For each unit pair $(A, B)$, for each non-unit production of the form $B \to \gamma$ in $P$, add the production $A \to \gamma$ in $P$.

**Algorithm 8.7** Given a CFG $G = (V, T, P, S)$ without $\epsilon$ productions and unit productions, the algorithm below converts all productions to productions of length 2 (involving only non-terminals on RHS) or productions of length 1 (involving only terminal on RHS).

1. For each production of the form $A \to X_1X_2 \cdots X_k$ for some symbols $X_1, X_2, \cdots, X_k$, replace it with the productions $A \to Z_1B_2, B_2 \to Z_2B_3, \cdots B_{k-1} \to Z_{k-1}Z_k$ where $B_i$ are new non-terminals. If $X_i \in T$, then $Z_i$ are new non-terminals and the production $Z_i \to X_i$ is added. Otherwise, $Z_i = X_i$.

**Algorithm 8.8** Given a CFG $G$ such that $\epsilon \notin L(G)$, the algorithm below converts $G$ into Chomsky Normal Form.

1. Remove $\epsilon$ productions from $G$.
2. Remove unit productions from $G$.
3. Convert all remaining productions to productions of length 2 (involving only non-terminals on RHS) or productions of length 1 (involving only terminal on RHS).

**Theorem 8.9** For any parse tree of a derivation of a string $w$ using a grammar in Chomsky Normal Form, if height of the parse tree is $s$, then $|w| \leq 2^{s-1}$.

**Theorem 8.10 (Pumping Lemma)** Let $L$ be a CFL. Then, there exists $n \in \mathbb{Z}^+$ such that for all $z \in L$ satisfying $|z| \geq n$, we can write $z = uvwxy$ such that

1. $|vwx| \leq n$
2. $vx \neq \epsilon$
3. For all $i \in \mathbb{N}$ we have $uv^iwx^iy \in L$

**Substitution** A mapping $s : \Sigma^* \to CFL$ is a substitution on the alphabet $\Sigma$ if and only if $s(\epsilon) = \{\epsilon\}$ and for all $a \in \Sigma$ and $w \in \Sigma^*$ we have $s(wa) = s(w)s(a)$.

**Theorem 8.11** For any substitution $s$ on $\Sigma$, if $L$ is a CFL over $\Sigma$, then $\cup_{w \in L} s(w)$ is a CFL.

**Theorem 8.12** If $L$ is context-free, then $L^R$ is context-free.

**Theorem 8.13** If $L$ is context-free and $R$ is regular, then $L \cap R$ is context-free.

**Theorem 8.14** A CFL $L$ is empty if and only if a CFG describing $L$ has the start symbol as a useless symbol.

**Algorithm 8.15 (CYK Algorithm)** Given a CFG $G = (V, T, P, S)$ and a string $w = a_1 a_2 \cdots a_n$ over $T$, the dynamic programming algorithm below determines whether $w \in L(G)$ by computing $X_{i,j} = \{A \in V : A \Rightarrow^* a_i a_{i+1} \cdots a_j\}$.

1. Let $X_{i,i} = \{A \in V :$ there is a production of the form $A \to a_i$ in $P\}$.
2. For $s = 1$ to $n - 1$, for $i = 1$ to $n - s$, let $j = i + s$, then let

$$X_{i,j} = \{A \in V : \exists B \in X_{i,k}, C \in X_{k+1,j} \text{ and a production of the form } A \to BC \text{ in } P\}$$

3. $w \in L(G)$ if and only if $S \in X_{1,n}$.

# Tutorial 4

**Question 2** A CFG $(V, T, P, S)$ is left-linear if and only if all its productions are of the form $A \to Bw$ or $A \to w$ for some $A, B \in V$ and $w \in T^*$. There is a right-linear grammar that describes $L$ if and only if there is a left-linear grammar that describes $L^R$. Moreover, there is a left-linear grammar that describes $L$ if and only if $L$ is regular.

# Tutorial 5

**Question 2** The presence of any memory device can sometimes reduce the number of states required to accept a regular language.

**Question 3** A two stack NPDA $NPDA = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, Z_0, Y_0, F)$ is a finite automaton where

- $Q$ is a finite set of states
- $\Sigma$ is a finite set of input symbols
- $\Gamma_1$ is a finite set of stack symbols for the first stack
- $\Gamma_2$ is a finite set of stack symbols for the second stack
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma_1 \times \Gamma_2 \to \mathcal{P}(Q \times \Gamma_1^* \times \Gamma_2^*)$ is a transition function
- $q_0 \in Q$ is the starting state
- $Z_0 \in \Gamma_1$ is the initial stack symbol on the first stack
- $Y_0 \in \Gamma_2$ is the initial stack symbol on the second stack
- $F \subseteq Q$ is a finite set of accepting states

Additionally, an instantaneous description of a two stack NPDA is a tuple $(q, w, \alpha, \beta)$ for some $q \in Q$, $w \in \Sigma^*$, $\alpha \in \Gamma_1^*$ and $\beta \in \Gamma_2^*$. Moreover, if $(p, \alpha', \beta') \in \delta(q, a, X, Y)$, then for any $w \in \Sigma^*$, $\alpha \in \Gamma_1$ and $\beta \in \Gamma_2$, we have $(q, aw, X\alpha, Y\beta) \vdash (p, w, \alpha'\alpha, \beta'\beta)$. $\vdash^*$ is the reflexive and transitive closure of $\vdash$. For acceptance by final state, we have

$$L(NPDA) = \{w \in \Sigma^* : (q_0, w, Z_0, Y_0) \vdash^* (q_f, \epsilon, \alpha, \beta) \text{ for some } q_f \in F, \alpha \in \Gamma_1^* \text{ and } \beta \in \Gamma_2^*\}$$

In fact, there exists a language that can be accepted by a two stack NPDA but not by a one stack NPDA.

# Tutorial 6

**Question 5** Given a CFG $G = (V, T, P, S)$, the algorithm below determines whether $G$ is describing a finite language.

1. Convert $G$ into Chomsky Normal Form and remove useless symbols.
2. Construct a directed graph with vertex set $V$ and there is an edge from $A$ to $B$ if and only if there is a production of the form $A \to \alpha B \beta$ in $P$.
3. $G$ is describing a finite language if and only if the constructed directed graph is acyclic.

**Question 6** Given a CFG $G = (V, T, P, S)$, the algorithm below constructs $Unit(A) = \{B \in V : (B, A) \text{ is a unit pair}\}$ for all $A \in V$.

1. Construct a directed graph with vertex set $V$ and there is an edge from $A$ to $B$ if and only if there is a production $A \to B$ in $P$.
2. Let $Unit(A) = \{B \in V : \text{there exists a path from } A \text{ to } B \text{ in the constructed directed graph}\}$.

**Question 7** A CFG $(V, T, P, S)$ is in Greibach Normal Form if and only if each of its productions are of the form $A \to a\alpha$ for some $a \in T$ and $\alpha \in (V \cup T)^*$. For every non-empty CFL $L$, there is a CFG in Greibach Normal Form that describes $L - \{\epsilon\}$.

# Tutorial 7

**Question 3** For any language $L$ over $\Sigma$, if $L$ is context free, then $Prefix(L)$ is context-free. If $L$ is regular, then $Prefix(L)$ is context-free, where $Prefix(L) = \{x \in \Sigma^* : \exists y \in \Sigma^*(xy \in L)\}$.

**Question 4** Let $L$ be a CFL. Then, there exists $n \in \mathbb{Z}^+$ such that for all $z \in L$ satisfying $|z| \geq n$, if we mark at least $n$ positions in $z$ to be distinguished, we can write $z = uvwxy$ such that

- $vwx$ has at most $n$ distinguished positions
- $vx$ has at least one distinguished position
- For all $i \in \mathbb{N}$ we have $uv^iwx^iy \in L$