CS 530: Theory of Computation

Based on Sipser (second edition): Notes on Context-Free Languages

Definition 1 (Definition 2.2, page 102) *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, and*
4. *$S \in V$ is the start variables.*

*If u, v, and w are strings of variables and terminals, and $A \to w$ is a rule of the grammar, we say that uAv **yields** uwv, written $uAv \Longrightarrow uwv$. Say that u **derives** v, written $u \Longrightarrow^* v$ if $u = v$ or if a sequence $u_1, u_2, ..., u_k$ exists for $k \geq 0$ and*

$$u \Longrightarrow u_1 \Longrightarrow u_2 \Longrightarrow ... \Longrightarrow u_k \Longrightarrow v.$$

*The **language of the grammar** is $\{w \in \Sigma^* | S \Longrightarrow^* w\}$.*

Definition 2 *A language L is context-free if there exists a context-free grammar G such that L is the language of G.*

Definition 3 *A **parse tree** for a grammar G is defined recursively as follows. A node with no children and labeled with a symbol from Σ_ϵ is a parse tree. If the grammar has a rule $A \to w$, for $A \in V$ and $w \in (V \cup \Sigma)^*$, then a parse tree can be obtained as follows: make a new node, the root of the parse tree, labeled A, with $|w|$ children, and the i^{th} child is a node labeled w_i that is the root of a parse tree. There is one exception: if $w = \epsilon$, then the node labeled A has one child, that most be a node labeled ϵ.*

Definition 4 *A derivation of a string w in a grammar G is a* ***leftmost derivation*** *if at every step the leftmost remaining variable is the one replaced.*

Definition 5 (Definition 2.7, page 106) *A string w is derived* ***ambiguously*** *in context-free grammar G if it has two or more different leftmost derivations. Grammar G is* ***ambiguous*** *if it generates some string ambiguously.*

Definition 6 (Definition 2.8, page 107) *A context-free grammar is in* ***Chomsky normal form*** *if every rule is of the form*

$$A \to BC$$
$$A \to \alpha$$

where α is any terminal and A, B, and C are any variables – except that B and C may not be the start variable. In addition we permit the rule $S \to \epsilon$, where S is the start variable.

Theorem 7 (Theorem 2.9, page 107) *Any context-free language is generated by a context-free grammar in Chomsky normal form.*

Definition 8 (Definition 2.13, page 111) *A* ***pushdown automaton*** *is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where Q, Σ, Γ, and F are all finite sets, and*

1. *Q is the set of states,*
2. *Σ is the input alphabet,*
3. *Γ is the stack alphabet,*
4. *$\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ is the transition function,*
5. *$q_0 \in Q$ is the start state, and*

6. $F \subseteq Q$ *is the set of accept states.*

A pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ *computes as follows. We say that string* w *can bring* M *from state* $q \in Q$ *and stack string* $s \in \Gamma^*$ *to state* $q' \in Q$ *and stack* $s' \in \Gamma^*$ *(or that* M *can move from configuration* (q, s) *to configuration* (q', s')*) if* w *can be written as* $w = w_1 w_2 \dots w_m$*, where each* $w_i \in \Sigma_\epsilon$ *and sequences of states* $r_0, r_1, \dots, r_m \in Q$ *and strings* $s_0, s_1, \dots s_m \in \Gamma^*$ *exist that satisfy the following three conditions.*

1. $r_0 = q$ *and* $s_0 = s$.
2. *For* $i = 0, \dots, m-1$*, we have* $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$*, where* $s_i = at$ *and* $s_{i+1} = bt$ *for some* $a, b \in \Gamma_\epsilon$ *and* $t \in \Gamma^*$*. This condition states that* M *moves properly according to the state, stack, and next input symbol.*
3. $r_m = q'$ *and* $s_m = s'$.

M *accepts input* w *if there exist* $q' \in F$ *and* $s' \in \Gamma^*$ *such that* w *can bring* M *from* q_0 *and* $s = \epsilon$ *to* q' *and* s'*. The condition that* $q = q_0$ *and* $s = \epsilon$ *signifies that* M *starts out properly, in the start state and with an empty stack. The condition* $q' \in F$ *states that an accept state occurs at the input end. The strings* s_i *represent the sequence of stack contents that* M *has on the accepting branch of the computation.*

Theorem 9 (Theorem 2.20, page 115) *A language is context free if and only if some pushdown automaton recognizes it.*

Lemma 10 (Lemma 2.21, page 115) *If a language is context free, then some pushdown automaton recognizes it.*

Claim 11 (used in the proof of the previous Lemma) *Let* $A \in V$ *and* $x \in \Sigma^*$*. A generates* x *in the grammar* G *if and only if* x *can bring the PDA* P *from* q_{main} *with stack* A *to* q_{main} *with empty stack.*

Lemma 12 (Lemma 2.27, page 119) *If a pushdown automaton recognizes some language, then it is context free.*

Claim 13 (Claim 2.30, page 121) *If A_{pq} generates x, then x can bring P from p with empty stack to q with empty stack.*

Claim 14 (Claim 2.31, page 121) *If x can bring P from p with empty stack to q with empty stack, then A_{pq} generates x.*

Corollary 15 (Corollary 2.32, page 122) *Every regular language is context free.*

Theorem 16 (Theorem 2.34, page 123) ***Pumping lemma for context-free languages.*** *If A is a 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 following conditions:*

1. *for each $i \geq 0$, $uv^ixy^iz \in A$,*
2. *$|vy| > 0$, and*
3. *$|vxy| \leq p$.*

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