Context-Free Languages

As pointed out before, the prototypical context-free language is

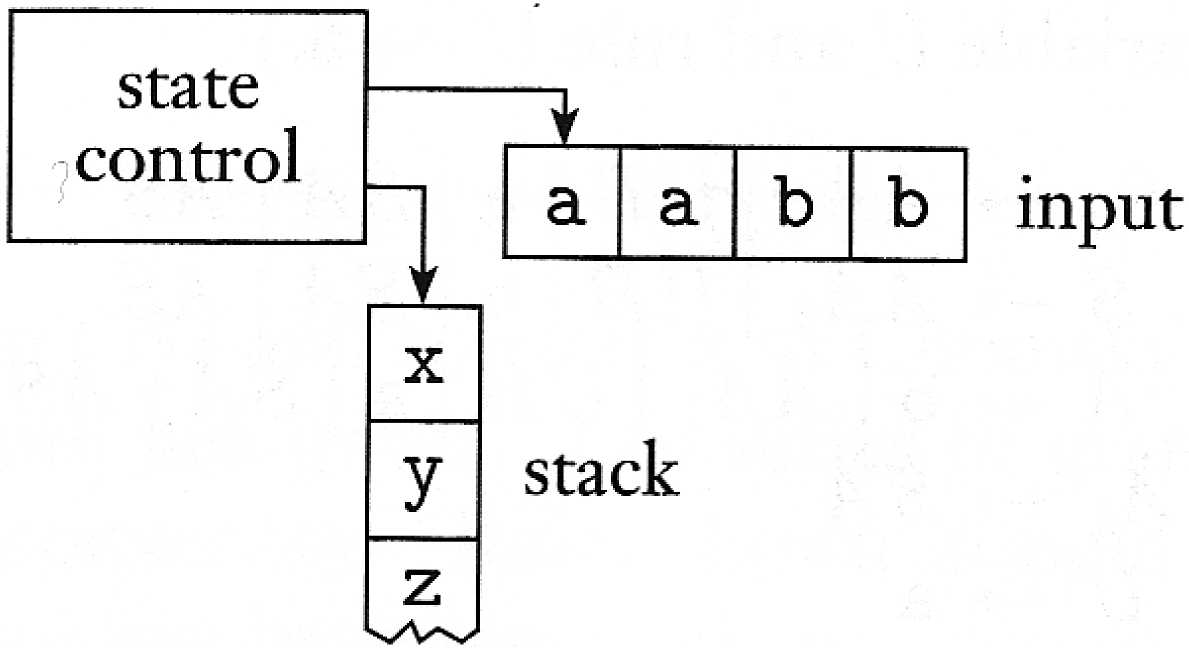
$$L = \{a^n b^n \mid a, b \in \Sigma \text{ and } n \geq 0\}$$

In order to accept strings in this language a machine has to remember how many a's it has seen so that it can match the number b's with the number of a's.

One way to accomplish this is with a stack, given some input string $s \in L$:

- ensure that only b's follow the last a in s,
- push all the a's of s onto the stack,
- then pop one a off the stack for each b,
- once we have read all the input symbols of s and the stack is empty and we are in an accepting state, then accept s; otherwise reject.

Pushdown Automaton



Formal Def. of PDA

Definition: a ***pushdown automaton*** is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

1. Q is the set of states,
2. Σ is input alphabet,
3. Γ is the stack alphabet,
4. $\delta : \ Q \times \Sigma_\epsilon \times \Gamma_\epsilon \longrightarrow P(Q \times \Gamma_\epsilon)$ is the transition function,
5. $q_0 \in Q$ is the start state, and
6. $F \subseteq Q$ is the set of accept states.

Is this a deterministic or nondeterministic machine?

Formal Computation of PDA

A pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows. It accepts input w if w can be written as $w = w_1 w_2 \cdots w_m$, where each $w_i \in \Sigma_\varepsilon$ and sequences of states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exist that satisfy the following three conditions. The strings s_i represent the sequence of stack contents that M has on the accepting branch of the computation.

1. $r_0 = q_0$ and $s_0 = \varepsilon$. This condition signifies that M starts out properly, in the start state and with an empty stack.
2. For $i = 0, \ldots, m-1$, 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 $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$. This condition states that M moves properly according to the state, stack, and next input symbol.
3. $r_m \in F$. This condition states that an accept state occurs at the input end.

$\{0^n1^n \mid n \geq 0\}$

The following is the formal description of the PDA (page 110) that recognize the language $\{0^n1^n \mid n \geq 0\}$. Let M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

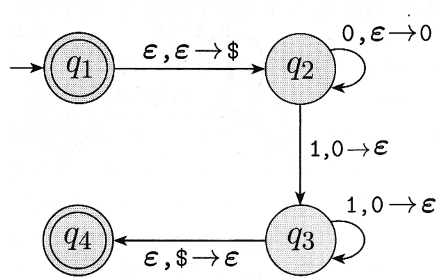
$$Q = \{q_1, q_2, q_3, q_4\},$$

$$\Sigma = \{0,1\},$$

$$\Gamma = \{0, \$\},$$

$F = \{q_1, q_4\}$, and

δ is given by the following table, wherein blank entries signify $\emptyset$.



Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$					
q_3				$\{(q_3, \varepsilon)\}$				$\{(q_4, \varepsilon)\}$	
q_4									

Context-Free Languages

Definition: A language is **context-free** if some pushdown automaton recognizes it.

Context-Free Grammars

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 variable.

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 $uAv \Rightarrow uwv$. Write $u \overset{*}{\Rightarrow} v$ if $u = v$ or if a sequence $u_1, u_2, \ldots, u_k$ exists for $k \geq 0$ and

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v.$$

The ***language of the grammar*** is $\{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$.

Context-Free Grammars

Example: Given the context-free grammar $G = (V, \Sigma, R, S)$, with $V = \{A\}, \Sigma = \{a, b\}, S = A$, and R the set of rules,

$$\begin{aligned} A &\rightarrow aAb \\ A &\rightarrow \epsilon \end{aligned}$$

then $L(G) = \{a^n b^n \mid n \geq 0\}$.

CFL Theorem

Theorem: A language is context-free iff some context-free grammar (CFG) generates it.

Proof Sketch:[a] Let L be some language.

If L is context-free, then some CFG generates it. If L is context-free then some PDA recognizes it. We can show that for every PDA we can build a CFG that generates the language the PDA recognizes.

If some CFG generates L, then L is context-free. For every CFG that generates L we can show that we can construct a PDA that recognizes L.

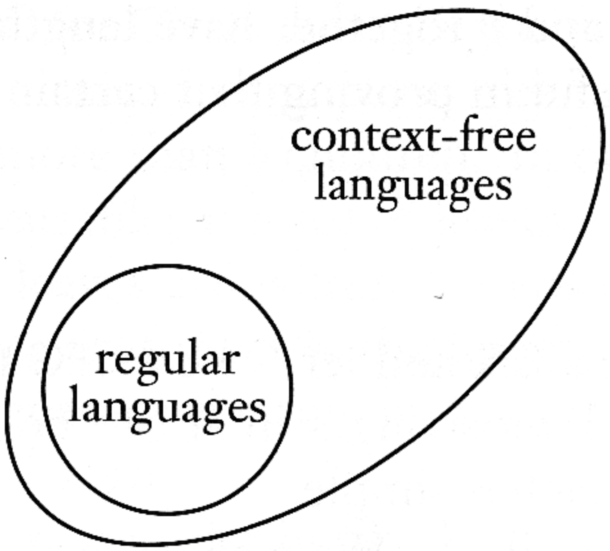
[a] A related formal proof appears in the book; pp106ff 1st ed., pp115ff 2nd ed.)

Language Hierarchy

Corollary: Every regular language is also a context-free language.

Proof Sketch: A PDA can simulate an FA by ignoring its stack.

This gives us the following hierarchy of languages



Chomsky Normal Form

The Chomsky normal form of a context free grammar is convenient to work with, especially later when we want to prove properties of context-free languages.

Definition: A context-free grammar (V, Σ, R, s) is in Chomsky normal form if every rule in R is of the form

$$
\begin{aligned}
A &\rightarrow BC \\
A &\rightarrow a
\end{aligned}
$$

with$a, B, C \in V$ and $a \in \Sigma$.

Chomsky Normal Form

Theorem: Any context-free language is generated by a context-free grammar in Chomsky normal form

Proof Sketch: Any context-free grammar can be converted to a grammar in Chomsky normal form.

Chomsky Normal Form

Example: Convert the following CFG to Chomsky Normal Form (CNF):

$$\begin{aligned} S &\rightarrow aX|Yb \\ X &\rightarrow S|\epsilon \\ Y &\rightarrow bY|b \end{aligned}$$

Step 1 - Kill all ϵ productions: By inspection, the only nullable nonterminal is X. Delete all ϵ productions and add new productions, with all possible combinations of the nullable X removed. The new CFG, without ϵ productions, is:

$$\begin{aligned} S &\rightarrow aX|a|Yb \\ X &\rightarrow S \\ Y &\rightarrow bY|b \end{aligned}$$

Chomsky Normal Form

Step 2 - Kill all unit productions: The only unit production is $X \rightarrow S$, where the S can be replaced with all SÕs non-unit productions (i.e. aX, a, and Yb). The new CFG, without unit productions, is:

$$\begin{aligned} S &\rightarrow aX|a|Yb \\ X &\rightarrow aX|a|Yb \\ Y &\rightarrow bY|b \end{aligned}$$

Step 3 - Replace all mixed strings with solid nonterminals. Create extra productions that produce one terminal, when doing the replacement. The new CFG, with a RHS consisting of only solid nonterminals or one terminal is:

$$\begin{aligned} S &\rightarrow AX|YB|a \\ X &\rightarrow AX|YB|a \\ Y &\rightarrow BY|b \\ A &\rightarrow a \\ B &\rightarrow b \end{aligned}$$

Beyond CFL's

Are there languages beyond context-free languages? Yes, consider

$$L = \{a^n b^n c^n \mid a, b, c \in \Sigma \text{ and } n \geq 0\}.$$

Our stack approach does not work anymore because we need to keep track of three entities.

CFL Pumping Lemma

Theorem: [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 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. $|vy| > 0$,
3. $|vxy| \leq p$.

CFL Pumping Lemma

As before we can use the pumping lemma to show that certain languages are *not* context-free.

Theorem: The language $A = \{a^n b^n c^n \mid n \geq 0\}$ is not context free.

Proof: Proof by contradiction using the pumping lemma. If the language is context free then there should be some string $s \in A$ with $|s| \geq p$ where p is the pumping length. Let $s = a^p b^p c^p$ be that string. The pumping lemma state that we can split up the string into $s = uvxyz$ such that $uv^i xy^i z \in A$ for all $i \geq 0$. Given conditions 2 and 3 of the pumping lemma this is clearly not possible. $\square$

CFL Pumping Lemma

Observations: Where does the pumping length come from? We cannot use a pumping length derived from the PDA because the PDA now contains an infinite structure making the argument of a forced looping given a string with the length of at least the number states difficult.

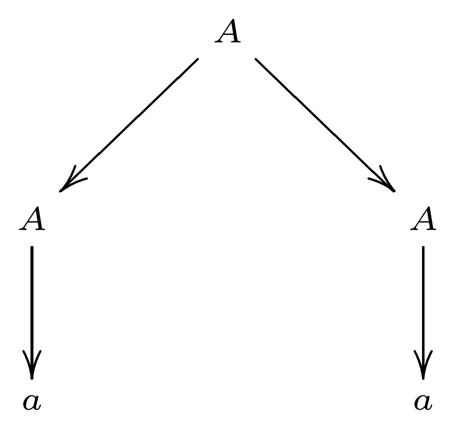
However, we can look at looping during derivations in grammars.

CFL Pumping Lemma

Consider the following grammar in Chomsky normal form of the language $L = \{a^n \mid n > 0\}$,

$$\begin{array}{ccc} A & \longrightarrow & AA \\ A & \longrightarrow & a \end{array}$$

The longest string we can generate without repeating a rule from the start symbol to a leaf node is 'aa',



That means, the derivation of any string with a length > 2 will force a *recursive* application of the first rule – that is, the derivation loops! But this would also mean that the associated PDA would loop!

CFL Pumping Lemma

Also notice that for a grammar in Chomsky normal form the length of a generated string s is related to the levels t in the derivation tree as

$$|s| \leq 2^t$$

We can relate this back to the number of non-terminals: Let V be the set of non-terminals in the grammar, then $|V|$ is the maximum number of levels in a parse tree without repeating a rule in any branch. Or in other words, if

$$|s| > 2^{|V|}$$

then one of its branches will have a repeated rule.

Check that in the grammar above.

CFL Pumping Lemma

Example: Consider the grammar in Chomsky normal form for the language

$$L = \{a^n b^n \mid n > 0\},$$

$$\begin{aligned} A &\rightarrow AP \\ P &\rightarrow QB \\ Q &\rightarrow AP \\ A &\rightarrow a \\ B &\rightarrow b \\ P &\rightarrow b \end{aligned}$$

Observe that $V = \{A, B, P, Q\}$, that is, strings with length $\geq 2^4$ will be generated using repeated rules.

CFL Pumping Lemma

Proposition: CFGs with recursive rules have a pumping length.

Proof: Follows directly from the fact that any CFG can be written in Chomsky Normal Form and that derivations in Chomsky Normal Form grammars are binary trees.

CFL Pumping Lemma

Example: Consider the CFG for the language

$$L = \{a^n b^n \mid n > 0\},$$

$$\begin{aligned} S &\longrightarrow ASB \\ S &\longrightarrow \epsilon \\ A &\longrightarrow a \\ B &\longrightarrow b \end{aligned}$$