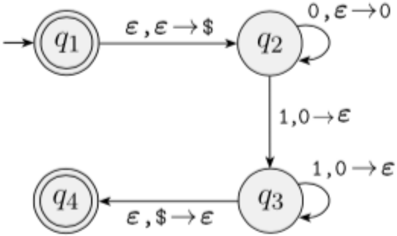
Administrivia

- Optional Exam #1: grades posted on Wednesday, 48 hours to review and comment
- Exercise sheet #3 assigned last Wednesday
- Reading: Chapter 2
- Last time: Pumping Lemma, Context Free Languages, Grammars
- Today: Push-down Automata, Grammars, Normal Forms

dog: Honey

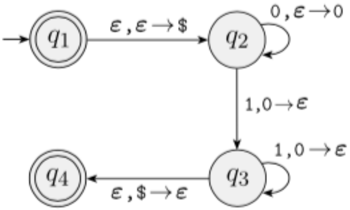
Context-Free Languages

- Regular languages are characterized by FAs and REs
- CFLs are characterized by **Context-Free Grammars (CFGs)**
- CFLs are characterized by **pushdown automata (PDAs)**
- PDAs are machines with memory (stack)
- Example of a CFL: $L = \{0^n1^n | n \geq 0\}$ → $\{0^n 1^{2n} \mid n \geq 0\}$
- Unlike FAs, DPDAs and NPDAs are not equivalent in power
- CFGs are Type 2 grammars in Chomsky's hierarchy



for every 0, push 2 0s on stack

PDA



A pushdown automaton is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_s, F)$, where

- Q is a finite set called the states,
- Σ is a finite input alphabet,
- Γ is a finite stack alphabet,
- $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to P(Q \times \Gamma_\epsilon)$ is the transition function,
- $q_s \in Q$ is the start state,
- $F \subseteq Q$ is the set of accept states.

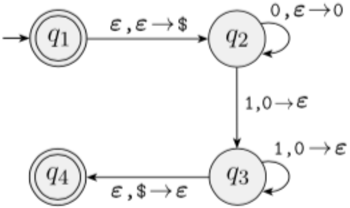
Computation of a Pushdown Automaton

Let $M = (Q, \Sigma, \Gamma, \delta, q_S, F)$ be a pushdown automaton and let $w = w_1 w_2 \ldots w_n$ be a string where each w_i is a member of the alphabet Σ_ϵ. Then M accepts w if there exists a sequence of states $r_0, r_1, \ldots, r_n$ in Q and a sequence of strings $s_0, s_1, \ldots, s_n$ in Γ_ϵ that satisfy the three conditions:

- $r_0 = q_S$ and $s_0 = \epsilon$
- $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, for $i = 0, ..., n-1$, where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\epsilon$ and $t \in \Gamma^*$.
- $r_n \in F$.

A language is called a **context free language** if some PDA recognizes it.

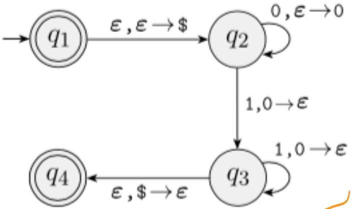
PDA Technicalities



The formal PDA definition has no explicit mechanism to test for:

- **Empty stack**. We accomplish this by initially placing a special symbol $ on the stack. When we see this on top of the stack we know that the stack is empty. This gives us the ability to "test for an empty stack."
- **End of input string**. We assume that the accept state takes effect only when the machine is at the end of the input, i.e., if the PDA is an accept state but the input has not yet been completely processed, the PDA does not yet accept.

PDA Technicalities



"$a, b \to b$"

on a, b b on top of stack,
leave b on top of stack

input character

top of stack

write on top of stack

The many meanings of $a, b \to c$: on a from input, pop b, push c:

- $a = \epsilon$: don't read from input
- $b = \epsilon$: don't pop top of stack
- $c = \epsilon$: don't push anything on top of stack
- $a, b \to \epsilon$: on a, pop b, (don't push)
- $a, \epsilon \to c$: on a (don't pop), push c

Let's Build a PDA for a Language

Consider the language $L = \{a^i b^j c^k | i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$

✓ ab   bc ✗

✗ aa   abc ✓✓

✓ ac

Let's Build a PDA for a Language

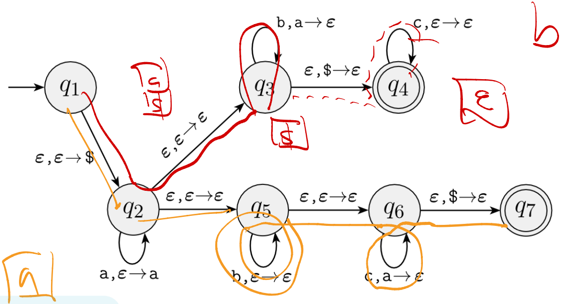
Consider the language $L = \{a^i b^j c^k | i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$

abbc

abcb

abc

cb still left



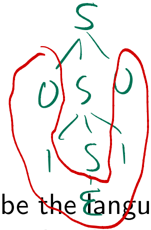
Recall Parse Trees

The parse trees for a grammar G are trees that must meet the following conditions:

$S \to 0S0 \to 01S10 \to 0110$

- each internal node v is labeled by a variable in $V \in G$
- each leaf node is labeled by either a terminal in Σ or ϵ; if a leaf is labeled with ϵ it is the only child of its parent
- if an internal node is labeled A and its children are $X_1, X_2, \ldots X_k$, then $A \to X_1 X_2 \ldots X_k$ is a rule in G

$S \to \epsilon | 0 | 1$
$S \to 0S0 | 1S1$



Another way to describe the language of a grammar is as the set of yields of the parse trees that have the start symbol as root and a terminal string as the yield.

Context Free Grammars and Normal Forms

AV → ... or BV→...

Definition

A context-free grammar is a 4-tuple (V, Σ, R, S), where:

1. V is a finite set of variables,
2. Σ is a finite set of terminals, $\Sigma \cap V = \emptyset$
3. R is a finite set of rules, $\alpha \to \beta$, where α is a variable and β a string of variables and terminals
4. $S \in V$ is the start variable

Example

1. $S \to ASA$
2. $S \to aB$
3. $A \to B$
4. $A \to S$
5. $B \to b$
6. $B \to \epsilon$

$$S \xrightarrow{1} ASA \xrightarrow{1} AASAA \xrightarrow{3} BASAA \xrightarrow{2} BBSAA \xrightarrow{3} BBSBA \xrightarrow{3} BBSBB \xrightarrow{2} BBaBBB \xrightarrow{6} BBaBB \xrightarrow{6} BaBB \xrightarrow{6} BaB \xrightarrow{6} aB \xrightarrow{5} a$$

Normal Forms for Type 2 Grammars

Simplified (standardized) rules such as those in the Chomsky Normal Form or the Greibach Normal Form, provide some nice properties of the derivations:

- the size of the generated string is proportional to the number of rules applied.
- Chomsky Normal Form (1959) makes it possible to use a polynomial time algorithm to decide whether a string can be generated by a grammar; see the Cocke-Younger-Kasami (CYK) algorithm.
- Greibach Normal Form (1965) makes it possible to prove that every CFG can be recognized by a PDA that works in real time (i.e., without ϵ transitions).

$A \rightarrow BC$
$A \rightarrow a$
$S \rightarrow \epsilon$ (if $\epsilon \in L$)

$A \rightarrow aA_1A_2 \dots A_k, k \geq 0$
$S \rightarrow \epsilon$ (if $\epsilon \in L$)

Chomsky Normal Form

Definition

A context-free grammar is in Chomsky normal form if every rule is of the form:

$A \rightarrow BC$

$A \rightarrow a$

where a 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 $S \rightarrow \epsilon$ if $\epsilon \in L$, where S is the start variable.

Clearly CNF and GNF are much more restricted than the initial definition of Context Free Grammars. So it is not obvious that they are as powerful (i.e., recognize the same class of languages)...

CFGs in CNF

Why do we care to force CFGs in a special form such as CNF?

unit

- rules of the type $A \to B$ are confusing: are we making any progress towards generating terminals?
- there could be loops in the derivation: $A \to B$, $B \to C$, $C \to A$
- rules such as $A \to \epsilon$ allow us to generate very long strings of variables and then erase them all
- Without "unit production rules" and without ϵ rules, every step in the derivation makes demonstrable progress toward the terminal string: either the string gets longer or a terminal appears.
- as you'll see in the exercises, deriving a word length n in the language takes exactly $2n - 1$ steps.
- finally, the parse tree is binary tree!

Any CFL has a CFG in CNF

Theorem

Any context-free language can be generated by a context-free grammar in Chomsky normal form.

Proof.

Let L be a CFL. Then there exists a type 2 grammar G for it. We show how to convert G into Chomsky normal form. The conversion has several stages wherein rules that violate the conditions are replaced with equivalent ones that are satisfactory.

1. add a new start variable.
2. eliminate all ϵ-rules of the form $A \to \epsilon$.
3. eliminate all unit rules of the form $A \to B$.
4. convert the remaining rules into the proper form. with extra variables

□

CFG to CNF Example

Phase 1 (new start)

$S \to ASA|aB$
$A \to B|S$
$B \to b|\epsilon$

valid CFG

$\mathbf{S_0} \to \mathbf{S}$
$S \to ASA|aB$
$A \to B|S$
$B \to b|\epsilon$

Phase 2 (remove $B \to \epsilon$)

$S_0 \to S$
$S \to ASA|aB$
$A \to B|S$
$B \to b|\epsilon$

$S_0 \to S$
$S \to ASA|aB|\mathbf{a}$
$A \to B|S|\boldsymbol{\epsilon}$
$B \to b$

Phase 2 (remove $A \to \epsilon$)

$S_0 \to S$
$S \to ASA|aB|a$
$A \to B|S|\epsilon$
$B \to b$

$S_0 \to S$
$S \to ASA|aB|a|\mathbf{SA}|\mathbf{AS}|\mathbf{S}$
$A \to B|S$
$B \to b$

CFG to CNF Example

Phase 3 (remove $S \rightarrow S$)

$S_0 \rightarrow S$

$\mathrm{S} \rightarrow ASA|aB|a|SA|AS|S$

$\mathrm{A} \rightarrow B|S$

$\mathrm{B} \rightarrow b$

$S_0 \rightarrow S$

$\mathrm{S} \rightarrow ASA|aB|a|SA|AS$

$\mathrm{A} \rightarrow B|S$

$\mathrm{B} \rightarrow b$

Phase 3 (remove $S_0 \rightarrow S$)

$S_0 \rightarrow S$

$\mathrm{S} \rightarrow ASA|aB|a|SA|AS$

$\mathrm{A} \rightarrow B|S$

$\mathrm{B} \rightarrow b$

$S_0 \rightarrow \mathbf{ASA|aB|a|SA|AS}$

$\mathrm{S} \rightarrow ASA|aB|a|SA|AS$

$\mathrm{A} \rightarrow B|S$

$\mathrm{B} \rightarrow b$

CFG to CNF Example

Phase 3 (remove $A \to B$)

$$\begin{aligned} S_0 &\to ASA|aB|a|SA|AS \\ \text{S} &\to ASA|aB|a|SA|AS \\ \text{A} &\to B|S \\ \text{B} &\to b \end{aligned}$$

$$\begin{aligned} S_0 &\to ASA|aB|a|SA|AS \\ \text{S} &\to ASA|aB|a|SA|AS \\ \text{A} &\to S|\mathbf{b} \\ \text{B} &\to b \end{aligned}$$

Phase 3 (remove $A \to S$)

$$\begin{aligned} S_0 &\to ASA|aB|a|SA|AS \\ \text{S} &\to ASA|aB|a|SA|AS \\ \text{A} &\to S|b \\ \text{B} &\to b \end{aligned}$$

$$\begin{aligned} S_0 &\to ASA|aB|a|SA|AS \\ \text{S} &\to ASA|aB|a|SA|AS \\ \text{A} &\to b|\mathbf{ASA}|\mathbf{aB}|\mathbf{a}|\mathbf{SA}|\mathbf{AS} \\ \text{B} &\to b \end{aligned}$$

CFG to CNF Example

Phase 4 (fix remaining rules)

$S_0 \rightarrow \underline{ASA}|a\overline{B}|a|SA|AS$
S $\rightarrow \underline{ASA}|aB|a|SA|AS$
A $\rightarrow b|\underline{ASA}|\overline{aB}|a|SA|AS$
B $\rightarrow b$

$S_0 \rightarrow aB$

$\Downarrow$

$S_0 \rightarrow UB$
$U \rightarrow a$

$S_0 \rightarrow ASA$

$\Downarrow$

$S_0 \rightarrow AA_1$
$A_1 \rightarrow SA$

$S_0 \rightarrow AA_1|UB|a|SA|AS$
S $\rightarrow AA_1|UB|a|SA|AS$
A $\rightarrow b|AA_1|UB|a|SA|AS$
$A_1 \rightarrow SA$
U$\rightarrow a$
B $\rightarrow b$

Ambiguity: Syntactic, Semantic, etc.

Context Free Languages can contain ambiguous sentences:

- Squad helps dog bite victim

Ambiguity: Syntactic, Semantic, etc.

Context Free Languages can contain ambiguous sentences:

- Squad helps dog bite victim
- MBA studies mushroom

Ambiguity: Syntactic, Semantic, etc.

Context Free Languages can contain ambiguous sentences:

- Squad helps dog bite victim
- MBA studies mushroom
- Enraged cow injures farmer with ax

Ambiguity: Syntactic, Semantic, etc.

Context Free Languages can contain ambiguous sentences:

- Squad helps dog bite victim
- MBA studies mushroom
- Enraged cow injures farmer with ax
- Kids make nutritious snacks

Ambiguity: Syntactic, Semantic, etc.

Context Free Languages can contain ambiguous sentences:

- Squad helps dog bite victim
- MBA studies mushroom
- Enraged cow injures farmer with ax
- Kids make nutritious snacks
- Hershey bars protest

Ambiguity: Syntactic, Semantic, etc.

Context Free Languages can contain ambiguous sentences:

- Squad helps dog bite victim
- MBA studies mushroom
- Enraged cow injures farmer with ax
- Kids make nutritious snacks
- Hershey bars protest
- Prostitutes appeal to pope

Ambiguity: Syntactic, Semantic, etc.

Context Free Languages can contain ambiguous sentences:

- Squad helps dog bite victim
- MBA studies mushroom
- Enraged cow injures farmer with ax
- Kids make nutritious snacks
- Hershey bars protest
- Prostitutes appeal to pope
- I never said she stole my money

syntactic

semantic

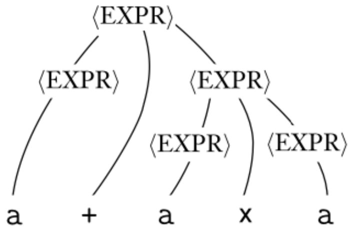
Ambiguity

Definition

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.

Consider the following grammar:

$< EXPR > \rightarrow < EXPR > + < EXPR > \mid < EXPR > x < EXPR > \mid (< EXPR >) \mid a$



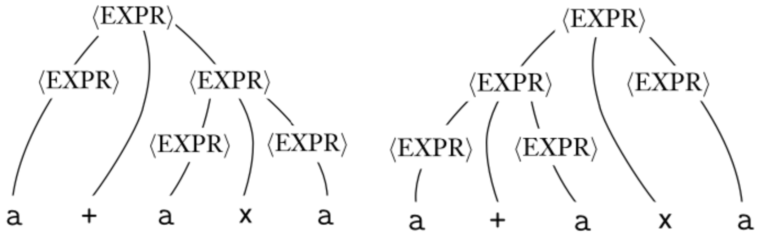
Ambiguity

Definition

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.

Consider the following grammar:

$< EXPR > \rightarrow < EXPR > + < EXPR > \mid < EXPR > x < EXPR > \mid (< EXPR >) \mid a$



Languages Recognized by PDAs and Generated by CFGs

Theorem

A language is context free if and only if some pushdown automaton recognizes it.

Proof.

Recall that by definition, a language is CF if some CFG generates it. Then we need to show that the class of CFLs is equivalent to the class of languages recognized by PDAs.

- $\Rightarrow$ If G is a CFG for language A, then there exists PDA P that recognizes A
- $\Leftarrow$ If PDA P recognizes a language A, then there exists a CFG G that generates A

□