CISC 4090
Theory of Computation

Context-Free Languages and Push Down Automata

Professor Daniel Leeds
dleeds@fordham.edu
JMH 332

Languages: Regular and Beyond

Regular:
- Captured by Regular Operations $(a \cup b) \cdot c^* \cdot (d \cup e)$
- Recognized by Finite State Machines

Context Free Grammars:
- Human language
- Parsing of computer language

2

An example Context-Free Grammar

Grammar G1
$A \rightarrow 0A1$
$A \rightarrow B$
$B \rightarrow \#$

Example strings generated:
#, 0#1, 00#11, 000#111, …

$L(G1) = \{0^n\#1^n \mid n \geq 0\}$

Variables: A, B; Terminals: 0, 1, #

One start variable: A

Substitution rules/productions
- Variable -> Variables, Terminals

3

Example English Grammar

Sentence -> NounPhrase VerbPhrase
NounPhrase -> Article NounSub
NounSub -> Noun | Adjective NounSub
VerbPhrase -> Verb | Verb NounPhrase
Noun -> Girl | Boy | Duck | Ball
Article -> The | A
Verb -> Throws | Sings

Example 1:
S -> NP VP
-> A NS V
-> A N V
-> The Boy Sings

Example 2:
S -> NP VP
-> A NS V
-> A N V
-> A Duck Throws

4

Formal CFG Definition

A CFG is a 4-tuple (V, Σ, R, S)
- V is finite set of variables
- Σ finite set of terminals
- R finite set of rules
- $S \in V$ start variable

5

Another example

$G3 = (\{S\}, \{a, b\}, R, S)$

R: $S \rightarrow aSb \mid SS \mid \varepsilon$

Example rule expansion:

S -> aSb	S -> SS
aaSbb	aSb aSb
aaεbb	aεb aaSbb
aabb	aεb aaεbb
	abaabb

Example strings generated:
ε, ab, abab, aabb, aaabbbab,
ababababab, abaaabbb, …

L(G3) = {a's & b's; each a is followed by a matching b, every b matches exactly one corresponding preceding a}
(like parenthesis matching)

7

Parenthesis-Math/Equation Grammar

$G = (\{S, A\}, \{(,), 0, \ldots, 9, +, *, -, /\}, R, S)$

R: $S \rightarrow (S) \mid SS \mid AS \mid \varepsilon$

$A \rightarrow 1|2|3|4|5|6|7|8|9|0| + \mid - \mid * \mid /$

8

Another example

$G4 = (\{A, B, C\}, \{a, b, c\}, R, A)$

R: $A \rightarrow aA \mid BC \mid \varepsilon$

$B \rightarrow Bb \mid C$

$C \rightarrow c \mid \varepsilon$

Example strings generated: ε, aaa, cbbc, aacc

L(G4) = {Hard to describe… }

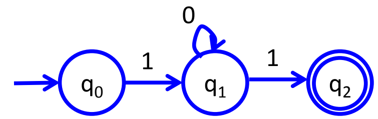
10

Designing CFGs

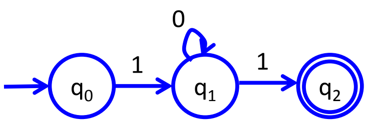
Creativity required

- If CFL is union of simpler CFL, design grammar for simpler ones (G1, G2, G3), then combine: S -> G1 | G2 | G3
- **If language is regular, can make CFG mimic DFA**

Example: express as CFG



Example: express as CFG



Q_0 -> $1Q_1$
Q_1 -> $0Q_1$ | $1Q_2$
Q_2 -> ε

Designing CFGs

Creativity required

- **If language is regular, can make CFG mimic DFA**
  - Match each state with a single corresponding variable
    - $Q=\{q_0,\ldots,q_n\}$ $V=\{R_0, \ldots, R_n\}$
  - Start state q_0 corresponds to state variable S -> R_0
  - Replace transition function with Production rule
    - $\delta(q_i, a) = q_j$ $R_i \rightarrow aR_j$
  - Accept state q_k : transition to ε $R_k \rightarrow \varepsilon$

Chomsky Normal Form

CFG is in Chomsky normal form if every rule takes form:

$\mathrm{A} \rightarrow \mathrm{BC}$

$\mathrm{A} \rightarrow \mathrm{a}$

- B and C may not be the start variables
- The start variable may transition to ε

Any CFL can be generated by CFG in Chomsky Normal Form

15

Converting to Chomsky Normal Form

- $S_0 \rightarrow S$ where S was original start variable
- Remove $A \rightarrow \varepsilon$
- Shortcut all unit rules
  Given $A \rightarrow B$ and $B \rightarrow u$, add $A \rightarrow u$
- Replace variable-terminal rules with variable-variable rules
  Given $A \rightarrow Bc$, add $U_C \rightarrow c$ and change A to $A \rightarrow BU_C$
- Replace rules $A \rightarrow u_1u_2u_3 \dots u_k$ with:
  $A \rightarrow u_1A_1,\ A_1 \rightarrow u_2A_2,\ A_2 \rightarrow u_3A_3,\ \dots,\ A_{k-2} \rightarrow u_{k-1}u_k$

16

Conversion practice

Non-normal form:

$S \rightarrow aSa|bX$

$X \rightarrow Ycc|\varepsilon$

$Y \rightarrow d|c$

17

Conversion practice

Non-normal form:

$S \rightarrow aSa|bX$

$X \rightarrow Ycc|\varepsilon$

$Y \rightarrow d|c$

Step 1: S_0->S,

$S_0 \rightarrow S$

$S \rightarrow aSa|bX$

$X \rightarrow Ycc|\varepsilon$

$Y \rightarrow d|c$

Step 2: Remove ε,

$S_0 \rightarrow S$

$S \rightarrow aSa|bX|b$

$X \rightarrow Ycc$

$Y \rightarrow d|c$

Step 3: Use unit rules,

$S_0 \rightarrow aSa|bX|b$

$S \rightarrow aSa|bX|b$

$X \rightarrow Ycc$

$Y \rightarrow d|c$

18

Conversion practice

Step 3: Use unit rules,

$S_0 \rightarrow aSa|bX|b$

$S \rightarrow aSa|bX|b$

$X \rightarrow Ycc$

$Y \rightarrow d|c$

Step 4: Replace terminals,

$S_0 \rightarrow ASA|BX|b$

$S \rightarrow ASA|BX|b$

$X \rightarrow YCC$

$Y \rightarrow d|c$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

Step 5: Reduce multi-variable

$S_0 \rightarrow AN|BX|b$

$S \rightarrow AN|BX|b$

$X \rightarrow YM$

$Y \rightarrow d|c$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

$N \rightarrow SA$

$M \rightarrow CC$

19

Ambiguity – examples

A grammar may generate a string in multiple ways

Math example:

Expr $\rightarrow$ Expr + Expr | Expr $\times$ Expr | Expr | a

English example:

the girl touches the boy with the flower

20

Ambiguity – definitions

A grammar generates a string ambiguously if there are two or more different parse trees

Definitions:

- Leftmost derivation: at each step the leftmost remaining variable is replaced
- *w* is derived **ambiguously** in CFG G if there exist more than one leftmost derivations

21

Conversion practice

Non-normal form:

$S \rightarrow aa|bXc$

$X \rightarrow Xc|Y$

$Y \rightarrow Ycc|a$

22

Conversion practice

Non-normal form:

$S \rightarrow aa|bXc$

$X \rightarrow Xc|Y$

$Y \rightarrow Ycc|a$

Step 1: Replace unit rules

$S \rightarrow aa|bXc$

$X \rightarrow Xc|Ycc|a$

$Y \rightarrow Ycc|a$

Step2: Replace terminals

$S \rightarrow AA|BXC$

$X \rightarrow XC|YCC|a$

$Y \rightarrow YCC|a$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

23

Conversion practice

Step2: Replace terminals

$S \rightarrow AA|BXC$

$X \rightarrow XC|YCC|a$

$Y \rightarrow YCC|a$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

Step 3: Reduce multi-var

$S \rightarrow AA|BN$

$X \rightarrow XC|YM|a$

$Y \rightarrow YM|a$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

$N \rightarrow XC$

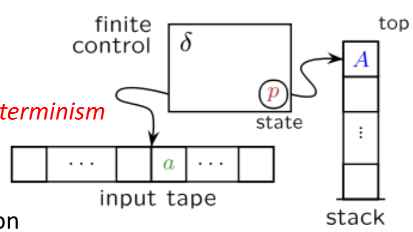
$M \rightarrow CC$

24

Push down automata

FSA augmented with memory

Equivalent to CFG *if use non-determinism*



Finite control: transition function

Tape: holds input string

Stack: Can write to/read from stack

Input is Last In First Out ("LIFO")

25

PDA and Language 0^n1^n

Read symbol from input, push each 0 onto stack

As soon as see 1's, start popping 0 for each 1 seen

- If finish reading and stack empty, accept
- If stack is empty and 1's remain, reject
- If inputs finished but stack still has 0's, reject
- In 0 appears on input, reject

26

Definition of PDA

A PDA is a 6-tuple $(\mathrm{Q}, \Sigma, \Gamma, \delta, \mathrm{q}_0, \mathrm{F})$ where Q, Σ, Γ, and F are finite sets

- Q is sets of states
- Σ is the input alphabet
- Γ is the stack alphabet
- $\delta\colon \mathrm{Q} \times \Sigma\varepsilon \times \Gamma\varepsilon \rightarrow \mathrm{P}(\mathrm{Q} \times \Gamma\varepsilon)$ is transition function
- $\mathrm{q}_0 \in \mathrm{Q}$ is start state
- $\mathrm{F} \subseteq \mathrm{Q}$ is set of accept states

PDA computation

M must start in q_0 with empty stack

M must move according to transition function

To accept string, M must be at accept state at end of input

Start stack with \$. If you see \$ at top of stack, it is empty

Understanding transition δ

$a, b \rightarrow c$ means:

- when you read a from tape and b is on top of stack
- replace b with c on top of stack

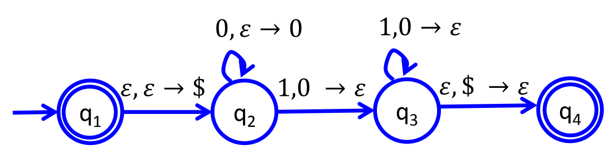
a, b, or c can be ε

- If a is ε then change stack without reading a symbol
- If b is ε then push new symbol c without popping b
- If c is ε then no new symbol pushed, only pop b

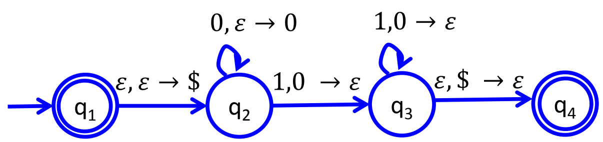
PDA to accept 0^n1^n

M1 is $(\mathrm{Q}, \Sigma, \Gamma, \delta, \mathrm{q}_0, \mathrm{F})$

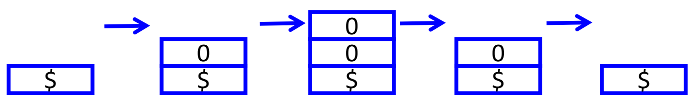
- $Q = \{q_1, q_2, q_3, q_4\}$ $\Sigma = \{0,1\}$
- $\Gamma = \{0, \$\}$ $\mathrm{F} = \{\mathrm{q}_1, \mathrm{q}_4\}$



PDA to accept 0^n1^n



Input: 0011

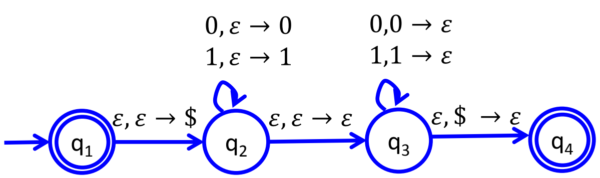


31

PDA to accept $\{ww^R\}$

Power of non-determinism:

- At start, don't know where string w ends

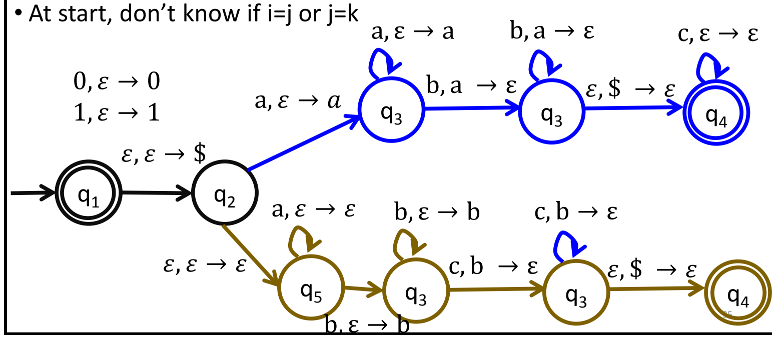


33

PDA to accept $a^ib^jc^k$, i=j or j=k

Power of non-determinism:

- At start, don't know if i=j or j=k



Theorem: A language is context free if and only if some PDA recognizes it

Let's prove: If a language L is CFL, some PDA recognizes it

Idea: Show how CFG can define a PDA

- Stack has set of terminals/variables to compare with input
- Place proper terminal/variable pattern onto stack based on rules
- Non-determinism: Clone your machine, following different branches of rules

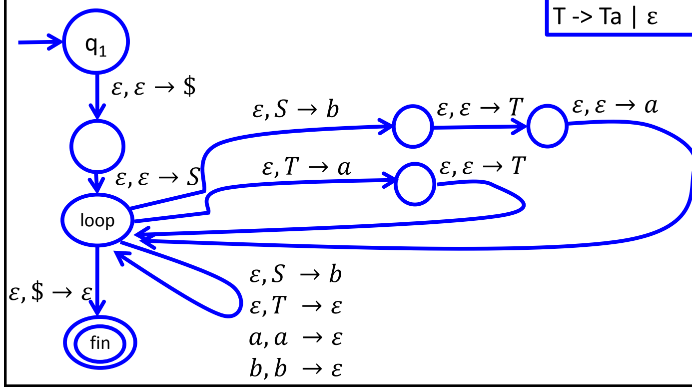
36

CFG -> PDA

- If top of stack is variable, sub one right-hand rule for the variable
- If top of stack is terminal, keep going iff terminal matches input
- If top of stack is $, accept!

Example 2.25 in textbook

S -> aTb | b
T -> Ta | ε



Regular languages vs. CFLs

- CFGs define CFLs
- PDAs recognize CFLs and Regular languages
- FSAs recognize Regular languages, but **not** CFLs
- CFLs and Regular languages not equivalent

Non Context Free Languages

Languages recognized by PDAs
- L={ww^R}
- L={a^nb^n | n≥0}

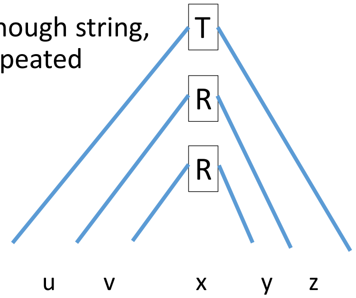
Languages **not** recognized by PDAs
- L={ww}
- L={$a^nb^nc^n$ | n≥0}

CFL pumping: Proof idea

Pigeonhole idea: Given a long enough string, some variable will need to be repeated

Example Grammar: S -> uRz

R -> x | vRy



Prove F={ww | w=$(0 \cup 1)^*$} not CFL

Try a sample string s={0^p10^p1} |s|>p

- Can we define uvxyz=s so $uv^ixy^iz \in F$?
- Yes: u=0^{p-1}, v=0, x=1, y=0, z=$0^{p-1}1$

Try another sample string s={$0^p1^p0^p1^p$}

- Can we define uvxyz=s so $uv^ixy^iz \in F$?
- No:
  - If vxy is in first w, pumping will make increase 1's and/or 0's in first w but not in second
  - If vxy straddles the middle, vxy will either increase 1's for first w and 0's for second w, or will break the 0^n1^n pattern