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: $(a \cup b) \cdot c^* \cdot b \cdot (d \cup e \cup a)$

Not-regular: $c^n b d^n$

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

3

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

Substitution rules/productions
- Variable -> Variables, Terminals

5

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

6

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

7

Another example

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

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

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

L(G1) = {a's & b's; each a is followed by a matching b, every b matches exactly one corresponding preceding a}
(like parenthesis matching)

8

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

Match each state with a single corresponding variable

$Q=\{q_0,\ldots,q_n\}$ $V=\{R_0, \ldots, R_n\}$

Replace transition function with Production rule

$\delta(q_i, a) = q_j$ $R_i \rightarrow aR_j$

Accept state q_k : transition to ε $R_k \rightarrow \varepsilon$

9

Ambiguity – examples

A grammar may generate a string in multiple ways

Math example:

$\text{Expr} \rightarrow \text{Expr} + \text{Expr} \mid \text{Expr} \times \text{Expr} \mid \text{Expr} \mid a$

English example:

the girl touches the boy with the flower

10

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

11

a+axa+a leftmost derivations

Derivation 1:	Derivation 2:
Expr	Expr
Expr x Expr	Expr + Expr
Expr + Expr x Expr	Expr x Expr + Expr
a + Expr x Expr	Expr + Expr x Expr + Expr
a + a x Expr	a + Expr x Expr + Expr
a + a x Expr + Expr	a + a x Expr + Expr
a + a x a + Expr	a + a x a + Expr
a + a x a + a	a + a x a + a

12

Chomsky Normal Form

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

$\mathrm{A} \to \mathrm{BC}$

$\mathrm{A} \to \mathrm{a}$

- 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

13

Converting to Chomsky Normal Form

- $S_0 \to S$ where S was original start variable
- Remove $A \to \varepsilon$
- For each multiple-occurrence of A, add new rules with A deleted
  $R \to uAvAw$ change to $R \to uvAw \mid uAvw \mid uvw$
- Shortcut all unit rules
  Given $A \to B$ and $B \to u$, add $A \to u$
- Replace rules $A \to u_1 u_2 u_3 \dots u_k$ with:
  $A \to u_1 A_1,\ A_1 \to u_2 A_2,\ A_2 \to u_3 A_3,\ \dots,\ A_{k-2} \to u_{k-1} u_k$

14

Conversion practice

Non-normal form:

$S \rightarrow aXbX$

$X \rightarrow aY|bY|\varepsilon$

$Y \rightarrow X|c$

Conversion practice, answer part 1

Non-normal form:

$S \rightarrow aXbX$

$X \rightarrow aY|bY|\varepsilon$

$Y \rightarrow X|c$

Step 1: S_0->S,
then place ε for X

$S_0 \rightarrow S$

$S \rightarrow aXbX|abX|aXb|ab$

$X \rightarrow aY|bY$

$Y \rightarrow \varepsilon|X|c$

Conversion practice, answer part 2

Step 1: S_0->S,
then place ε for X

$S_0 \rightarrow S$

$S \rightarrow aXbX|abX|aXb|ab$

$X \rightarrow aY|bY$

$Y \rightarrow \varepsilon|X|c$

Step 2: place ε for Y

$S_0 \rightarrow S$

$S \rightarrow aXbX|abX|aXb|ab$

$X \rightarrow aY|bY|a|b$

$Y \rightarrow X|c$

Conversion practice, answer part 3

Step 2: place ε for Y

$S_0 \rightarrow S$

$S \rightarrow aXbX|abX|aXb|ab$

$X \rightarrow aY|bY|a|b$

$Y \rightarrow X|c$

Step 3: replace S and X

$S_0 \rightarrow aXbX|abX|aXb|ab$

$S \rightarrow aXbX|abX|aXb|ab$

$X \rightarrow aY|bY|a|b$

$Y \rightarrow aY|bY|a|b|c$

Conversion practice, answer part 3

Step 3: replace S

$S_0 \to aXbX|abX|aXb|ab$

$S \to aXbX|abX|aXb|ab$

$X \to aY|bY|a|b$

$Y \to aY|bY|a|b|c$

Step 4: replace terminal-variable combos with variable combos

$S_0 \to U_1XU_2X|U_1U_2X|U_1XU_2|U_1U_2$

$S \to U_1XU_2X|U_1U_2X|U_1XU_2|U_1U_2$

$X \to U_1Y|U_2Y|a|b$

$U_1 \to a \quad U_2 \to b \quad Y \to U_1Y|U_2Y|a|b|c$

19

Conversion practice, answer part 3

Step 4: replace terminal-variable combos with variable combos

$S_0 \to U_1XU_2X|U_1U_2X|U_1XU_2|U_1U_2 \qquad U_1 \to a \quad U_2 \to b$

$S \to U_1XU_2X|U_1U_2X|U_1XU_2|U_1U_2 \qquad X \to U_1Y|U_2Y|a|b$

$Y \to U_1Y|U_2Y|a|b|c$

Step 4: replace multi-variables with pairs

$S_0 \to U_1C|U_1D|U_1E|U_1U_2$

$S \to U_1C|U_1D|U_1E|U_1U_2$

$C \to XD \quad D \to U_2X \quad E \to XU_2$

$X \to U_1Y|U_2Y|a|b$

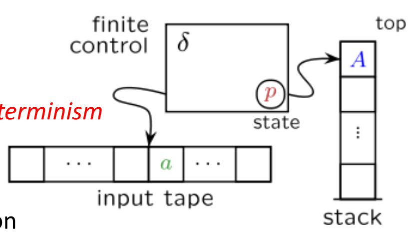
$U_1 \to a \quad U_2 \to b \quad Y \to U_1Y|U_2Y|a|b|c$

20

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")

21

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

22

Definition of PDA

A PDA is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, Γ, and F are finite sets

- Q is sets of states
- Σ is the input alphabet
- Γ is the stack alphabet
- $\delta: Q \times \Sigma\varepsilon \times \Gamma\varepsilon \rightarrow P(Q \times \Gamma\varepsilon)$ is transition function
- $q_0 \in Q$ is start state
- $F \subseteq Q$ is set of accept states

23

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

24

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

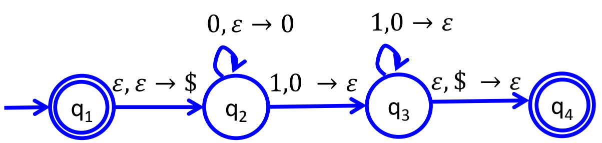
- If a is ε then change stack without reading a symbol
- If b is ε then push new symbol c without popping b
- If c is ε then no new symbol pushed, only pop b

25

PDA to accept 0^n1^n

M1 is $(Q, \Sigma, \Gamma, \delta, q_0, F)$

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

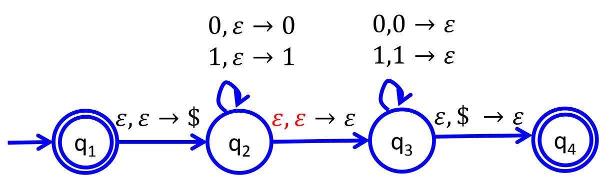


26

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

Power of non-determinism:

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

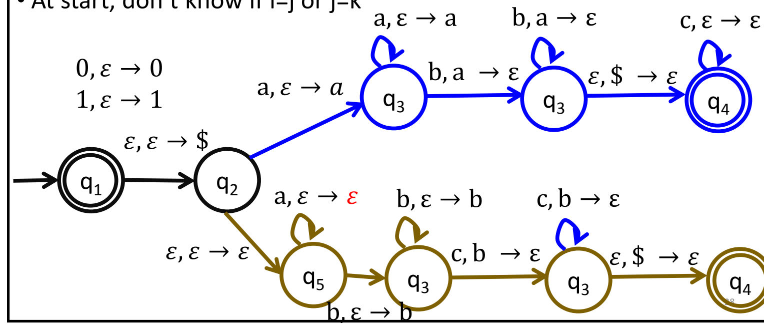


27

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

29

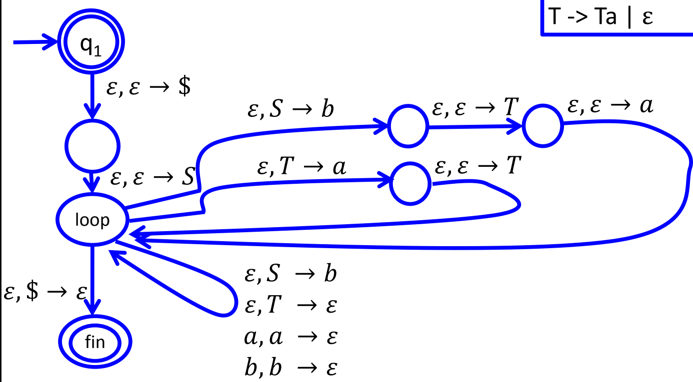
CFG -> PDA

- 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!

30

Example 2.25 in textbook

S -> aTb | b
T -> Ta | ε



31

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

32

Non Context Free Languages

Languages recognized by PDAs

- L={ww^R}
- L={$a^n b^n$ | n≥0}

Languages **not** recognized by PDAs

- L={ww}
- L={$a^n b^n c^n$ | n≥0}

33

Proving non context free – NEW pumping lemma!

Every string in CFL A with length greater than or equal to the pumping length *p* can be "pumped"

Every string $w \in A$ (|w|≥p) can be written as uvxyz where

1. For each $i \geq 0$, $uv^i xy^i z \in A$
2. |vy| > 0
3. $|vxy| \leq p$

34

Regular language PUMPING: Proof idea

If $|s| < p$, trivially true

If $|s| \geq p$, consider the states the FSA goes through

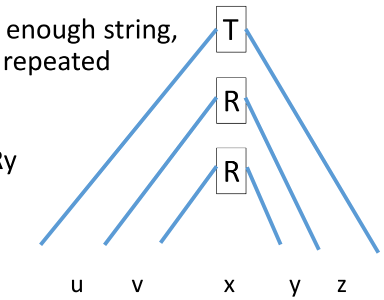
- Since there are only p states, $|s|>p$, one state must be repeated
- **Pigeonhole principle:** There must be a cycle

35

CFL pumping: Proof idea

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

Example Grammar: T -> uRz
R -> x | vRy



36

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

37

Prove $B=\{a^nb^nc^n \mid n \geq 0\}$ not CFL

38