INT201 Decision, Computation and Language

Lecture 7 – Context-Free Languages (3)

Dr Yushi Li and Dr Chunchuan Lyu

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Recap

- CFLs are closed under concatenation, union and Kleene closure
- CFLs/Natural language exhibits ambiguities (* optional)
- Pushdown automata are NFA with a stack

A PDA ***recognizes*** a language iff for all strings in the language, there ***exists one branch*** of computation that accepts it, and ***no other strings*** can be accepted by the PDA.

Today

- Equivalence between PDA and CFL
- Pumping Lemma for CFL

Equivalence of PDA and CFGs

Let Σ be an alphabet and let $A \subseteq \Sigma^*$ be a language. Then A is context-free if and only if there exists a nondeterministic pushdown automaton that recognizes A.

- If $A = L(G)$ for some CFG G, then $A = L(M)$ for some PDA M.
- If $A = L(M)$ for some PDA M, then $A = L(G)$ for some CFG G.

Proof If $A = L(G)$ for some CFG G, then $A = L(M)$ for some PDA M.

Basic idea: Given CFG G, convert it into PDA M with $L(M) = L(G)$ by building PDA that simulates a derivation.

Equivalence of PDA and CFGs

Convert CFG into PDA

CFG G= (V, Σ, R, S)

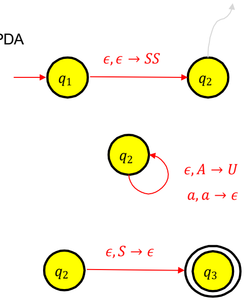
Start symbol S

Production rules $A \in V \rightarrow U \in (V \cup \Sigma)^*$

Terminal Symbols $a \in \Sigma$

End generation when no variables left.

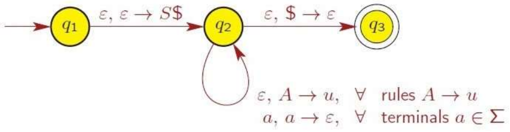
PDA



Problem: U could consist of multiple symbols, PDA only push one at a time.
Solution: Enter intermediate states when a production rule is applied, and use epsilon transitions to add symbols one by one.

Equivalence of PDA and CFGs

Convert CFG into PDA



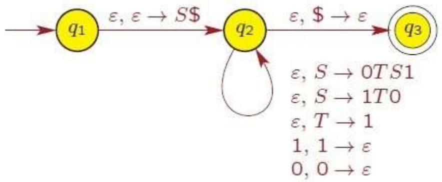
PDA works as follows:
1. Pushes \$ and then S on the stack, where S is start variable.
2. Repeats following until stack empty
(a) If top of stack is variable $A \in V$, then replace A by some $u \in (\Sigma \cup V)^*$, where $A \to u$ is a rule in R.
(b) If top of stack is terminal $a \in \Sigma$ and next input symbol is a, then read and pop a.
(c) If top of stack is \$, then pop it and accept.

Equivalence of PDA and CFGs

Example

Given CFG $G = (V, \Sigma, R, S)$
Variables $V = \{S, T\}$
Terminals $\Sigma=\{0, 1\}$
Rules: $S \to 0TS1 \mid 1T0$, $T \to 1$

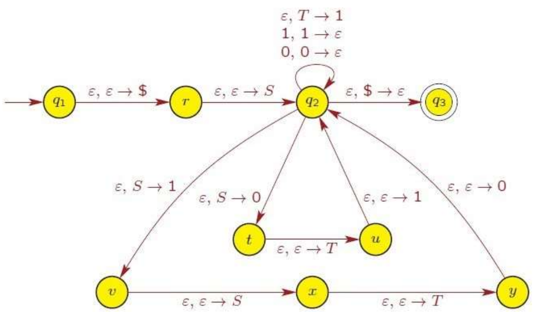
Convert the above CFG into a PDA.



PDA cannot push strings (e.g, S\$, 0TS1, 1T0) onto stack. We need to create the intermediate states.

Equivalence of PDA and CFGs

Example



Equivalence of PDA and CFGs

Convert PDA into CFG

We observe that the PDA we built from CFG looks quite special. It has a single accepting state with empty stack. This suggests that we need to somehow convert all PDA to some simplified PDA.

We propose the following simplification of PDA
1. PDA P has a single accepting state q_{accept},
2. PDA P accepts on empty stack,
3. Each transition either pushes a symbol onto the stack (a *push* move) or pops one off the stack (a *pop* move), but it does not do both at the same time.

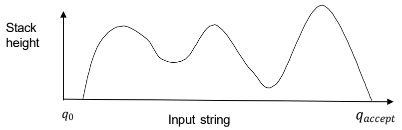
Lemma:
PDA with the above simplifications are equivalent to standard PDA.

Equivalence of PDA and CFGs

Convert PDA into CFG

1. PDA P has a single accepting state q_{accept},
2. PDA P accepts on empty stack,
3. Each transition either pushes a symbol onto the stack (a *push* move) or pops one off the stack (a *pop* move), but it does not do both at the same time.

Consequence of those simplifications:
The computation history of PDA will grow and shrink the stack.
We would like to construct CFG with variables A_{pq} generating all strings that take PDA from p with empty stack to q with empty stack (i.e., p to q without touching the stack below).
Intuitively, the empty stack condition allows CFG modeling.



Equivalence of PDA and CFGs

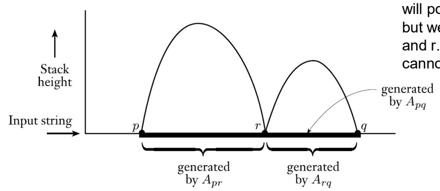
Convert PDA into CFG

A_{pq} generating all strings that take PDA from p with empty stack to q with empty stack (i.e., p to q without touching the stack below).

Formally, for PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{\text{accept}}\})$
We construct G= $(V, \Sigma, R, A_{q_0 q_{\text{accept}}})$ with $V = \{A_{pq} \mid p, q \in Q\}$, $\Sigma = \Sigma$
and most importantly production rules in three parts:

1. For each $p \in Q$, put $A_{pp} \to \epsilon$ (the base case)
2. For each $p, q, r \in Q$, put $A_{pq} \to A_{pr} A_{rq}$

Conditioning on the stack will pop out the element, but we are pushing at p and r. Remember that we cannot push and pop.



Equivalence of PDA and CFGs

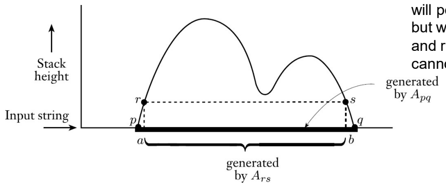
Convert PDA into CFG

A_{pq} generating all strings that take PDA from p with empty stack to q with empty stack (i.e., p to q without touching the stack below).

Production rules in three parts:

1. For each $p \in Q$, put $A_{pp} \to \epsilon$ (the base case)
2. For each $p, q, r \in Q$, put $A_{pq} \to A_{pr} A_{rq}$
3. For each $p, q, r, s \in Q$, $u \in \Gamma$ (stack symbol), and $a, b \in \Sigma_\epsilon$, if $\delta(p, a, \epsilon)$ contains (r, u) and $\delta(s, b, u)$ contains (q, ϵ), put $A_{pq} \to a\ A_{rs} b$

Conditioning on the stack will pop out the element, but we are pushing at p and r. Remember that we cannot push and pop.



Equivalence of PDA and CFGs

Convert PDA into CFG

A_{pq} generating all strings that take PDA from p with empty stack to q with empty stack (i.e., p to q without touching the stack below).

Production rules in three parts:

1. For each $p \in Q$, put $A_{pp} \to \epsilon$ (the base case)
2. For each $p, q, r \in Q$, put $A_{pq} \to A_{pr} A_{rq}$
3. For each $p, q, r, s \in Q$, $u \in \Gamma$ (stack symbol), and $a, b \in \Sigma_\epsilon$, if $\delta(p, a, \epsilon)$ contains (r, u) and $\delta(s, b, u)$ contains (q, ϵ), put $A_{pq} \to a\ A_{rs} b$

Formally, we still need to show this construction works in the sense that A_{pq} generates x if and only if x can bring PDA from p with empty stack to q with empty stack.

See section 2.2 in Sipser 3rd for proof based on mathematical induction on the length of x.

CFLs and regular languages

If A is a regular language, then A is also a CFL.

Proof

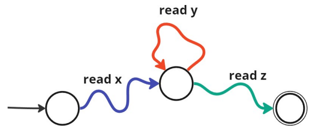
- Suppose A is regular, so it has a corresponding NFA.
- NFA is a PDA without stack.
- So A has a PDA
- A is context-free

Converse is not ture. (e.g, $L = \{ 0^n 1^n \mid n \geq 0 \}$ is CFL but not regular.)

Recap: pumping Lemma for RE

Let L be a regular language. Then there exists an integer $p \geq 1$, called the pumping length, such that the following holds: Every string s in L, with $|s| \geq p$, can be written as $s = xyz$, such that

1. $|y| \geq 1$ (i.e., v and y are not both empty),
2. $|xy| \leq p$, and
3. $xy^iz \in L$, for all $i \geq 0$.

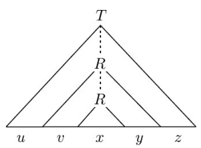


Pumping Lemma for CFLs

Let L be a context-free language. Then there exists an integer $p \geq 1$, called the pumping length, such that the following holds: Every string s in L, with $|s| \geq p$, can be written as $s = uvxyz$, such that

1. $|vy| \geq 1$ (i.e., v and y are not both empty),
2. $|vxy| \leq p$, and
3. $uv^ixy^iz \in L$, for all $i \geq 0$.

- Since the grammar is finite, if every variable only appear once in the parse tree, only strings of finite length can be generated. For long enough strings, there must exists variables being used at least twice.



Pumping Lemma for CFLs

Proof

Let b = the length of the longest right hand side of a rule (E → E+T)
= the max branching of the parse tree

Let h = the height of the parse tree for s.

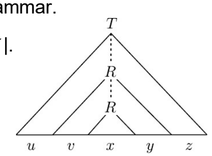
A tree of height h and max branching b has at most b^h leaves.
So $|s| \leq b^h$.

Let $p = b^{|V|} + 1$ where $|V|$ = # variables in the grammar.

So if $|s| \geq p > b^{|V|}$ then $|s| > b^{|V|}$ and so $h > |V|$.

As only variables can be nonterminal,

at least $|V| + 1$ variables occur in the longest path.
So some variable R must repeat on a path.

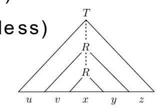


Pumping Lemma for CFLs

Proof

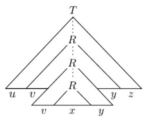
Every string s in L, with $|s| \geq p$, can be written as $s = uvxyz$, such that

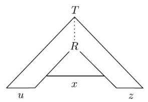
1. $|vy| \geq 1$ (CNF forbids ε of non-starting variable),
2. $|vxy| \leq p$, (repeat the construction if $|vxy|>p$)
3. $uv^ixy^iz \in L$, for all $i \geq 0$. (pump for more or less)



What if no long string?

The statement is vacuously true.





Pumping Lemma for CFLs

Example

Let's look at the palindrome CFL, $S \to \varepsilon \mid 0S0 \mid 1S1$

Branching factor	b=3
Varible numbers	V=1
Pumping length	$p = b^{\|V\|} + 1$=4

For strings of length 4, let's consider
s=0110
Pumping lemma claims
s = uvxyz, such that

1. |vy| ≥ 1 (i.e., v and y are not both empty),
2. |vxy| ≤ p, and
3. $uv^ixy^iz \in L$, for all i ≥ 0.

A. u=0,v=1,x= ε,y=1, z = 0
B. u=0,v= ε,x= 1 ,y=1, z = 0
C. u=01,v= ε,x= ε,y=1, z = 0
D. u=0,v= 1,x= ε,y= ε,z = 10

?

How to prove non-CFLs using pumping lemma

Example

Every string s in L, with $|s| \geq p$, can be written as $s = uvxyz$, such that

1. $|vy| \geq 1$ (i.e., v and y are not both empty),
2. $|vxy| \leq p$, and
3. $uv^ixy^iz \in L$, for all $i \geq 0$.

$L = \{a^n b^n c^n \mid n\}$ is non-CFL.

Proof by contradiction:

We assume that B is a CFL. Let p be the pumping length
$s = a^p b^p c^p$. Clearly s is a member of B and of length at least p.

We show that no matter how we divide s into $uvxyz$, one of the three conditions of the lemma is violated.

How to prove non-CFLs using pumping lemma

Example

Every string s in L, with $|s| \geq p$, can be written as $s = uvxyz$, such that

1. $|vy| \geq 1$ (i.e., v and y are not both empty),
2. $|vxy| \leq p$, and
3. $uv^ixy^iz \in L$, for all $i \geq 0$.

$L = \{a^n b^n c^n \mid n\}$ is non-CFL.

Proof by contradiction:

$$s = a^p b^p c^p$$

Condition 2 requires we find the vxy substring to be of length p at most.

Case 1: if a^{p-k}= vxy where $k \geq 0$, and due to $|vy| \geq 1$, pumping will change the number of a , and we get strings not in the language. Similarly, b^{p-k}= vxy or c^{p-k}= vxy won't work.

Now, vxy has to be a combination of two alphbets.

How to prove non-CFLs using pumping lemma

Example

Every string s in L, with $|s| \geq p$, can be written as $s = uvxyz$, such that

1. $|vy| \geq 1$ (i.e., v and y are not both empty),
2. $|vxy| \leq p$, and
3. $uv^ixy^iz \in L$, for all $i \geq 0$.

$\mathrm{L} = \{a^n b^n c^n | \text{ n}\}$ is non-CFL.

Proof by contradiction:

$$s = a^p b^p c^p$$

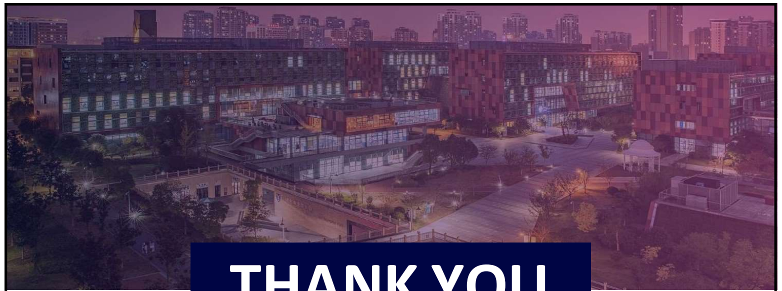
Condition 2 requires we find the vxy substring to be of length p at most.

Case 2: $a^m b^{p-m-k}$= vxy where $k \geq 0$. Still, after pumping either the number of a or b will be different from number of c. Similarly, $b^m c^{p-m-k}$= vxy won't work.

In all, we get a contradiction by assuming L is CFL, so it must not be.

Quick review

- CFG and PDA are equivalent
- There exists non context-free language provable by using the pumping lemma for CFLs.



THANK YOU

Xi'an Jiaotong-Liverpool University 西交利物浦大學

XJTLU | SCHOOL OF FILM AND TV ARTS