计算理论
Theory of Computation

https://courses.zju.edu.cn/course/join/5YB41KV75LZ

第3章 上下文无关语言
Context-free Language

杨莹春

yyc@zju.edu.cn

计算理论(周五第...

该群属于“浙江大学”内部群，仅组织内部成员可以加入，如果组织外部人员收到此分享，需要先申请加入该组织。

浙江大学曹光彪西楼-201

2021年10月21日

内容安排

– 3 classes （9/16,9/23,9/30 ）
  Sets, Relations and Language （CH1）
– 3 classes(10/7,10/14,10/21)
  Regular Language and Finite Automata (CH2)
– **3 classes(10/21**,10/28,11/4**)**
  Context-free Languages (CH3)
– 1 class(11/11) Mid-Term Review
– 3 classes (11/18,11/25,12/2)
  Turing machine (CH4)
– 2 classes(12/9,12/16)
  Undecidability （CH5）
– 2 classes(12/23,12/ 30) Final Review

Exam:2022/1

计算理论

第3章 上下文无关语言

Ch3． Context-free Language

Keywords III

- Ch3.　Context-free Languages

Context-free grammars, Pushdown automata, Equivalence of Pushdown automata and context-free grammars, Languages that are and are not context-free

Keywords III

- Ch3. Context-free Languages

Context-free grammars, Pushdown automata, Equivalence of Pushdown automata and context-free grammars, Languages that are and are not context-free

- Ch2. Regular Language and Finite Automata

Finite automata and Nondeterministic finite automata, Equivalence of finite automata and regular expressions, Languages that are and are not regular

Homework 3上交时间: 2020/12/25

Homework 3:	
P120	3.1.3(c) 3.1.9 (a)(b)
P135	3.3.2 (c)(d)
P142	3.4.1
P148	3.5.1 (b)(c)(d) 3.5.2 (c) 3.5.14 (a)(b)(c)(d) 3.5.15

Goal:
— to define increasingly powerful models of computation, more and more sophisticated devices for

- accepting languages
- generating languages

The Chomsky hierarchy

Language type	Automata type
regular	finite
context-free	pushdown
context-sensitive	linear bounded
unrestricted	Turing Machine

- **So-far: regular languages**
  - DFA = NFA (language recognizer)
  - Regular expressions(language generator)
  - Many languages are not regular
    - *Balanced parentheses*
    - *Arithmetic expressions*

- **Next: context-free languages**
  - PDA = CFG
  - Add LIFO (stack) memory

3.1 Context-Free Grammars

Example:

Consider the regular expression $a(a^* \cup b^*)b$.

- Regular expressions can be viewed as language generators.

Context-Free Grammar

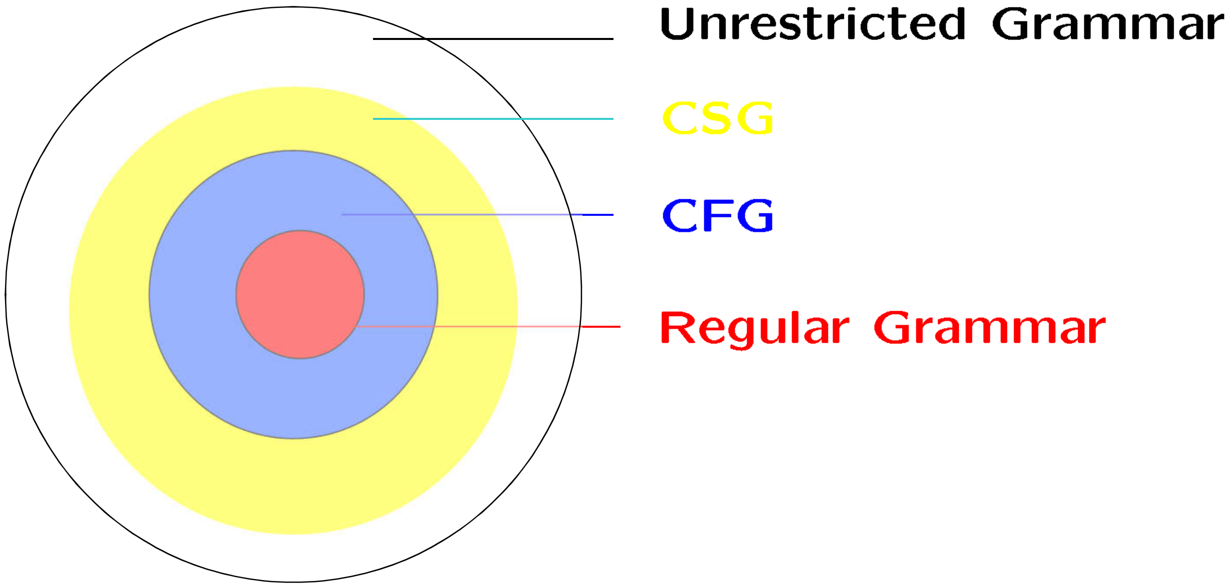
- The language denoted by $a(a^* \cup b^*)b$ can be defined by the following generator:

$$\boxed{\begin{array}{c} S \to aMb, M \to A, M \to B, \\ A \to aA, A \to e, B \to bB, B \to e. \end{array}}$$

Question: What is the context-free?

(as opposed to **Context-Sensitive Grammar**, CSG.)

Chomsky Hierarchy



Definition: A **context-free grammar(CFG)** is a quadruple $G = (V, \sum, R, S)$, where

V is an alphabet;

$\sum \subseteq V$ is the set of terminal symbols;

$S \in V - \sum$ is the start symbol; and

R is the set of rules, a finite subset of $(V - \sum) \times V^*$.

Remark:

- The member of $V - \sum$ are called **nonterminals**.
  For any $A \in V - \sum$ and $u \in V^*$,

$$A \rightarrow_G u \Leftrightarrow (A, u) \in R.$$

- For any strings $u, v \in V^*$,

$$u \Rightarrow_G v \Leftrightarrow \exists x, y \in V^*, \text{ and } A \in V - \sum, \text{ such that}$$
$$u = xAy, v = xv'y, \text{ and } A \rightarrow_G v'.$$

- $\Rightarrow_G^*$ is the reflexive, transitive closure of $\Rightarrow_G$.

- $w_0 \Rightarrow_G w_1 \Rightarrow_G \cdots \Rightarrow_G w_n$

  — a **derivation** in G of w_n from w_0. n be the length of the derivation.

- The **language generated by** G,

$$L(G) = \{w \in \sum^* : S \Rightarrow_G^* w\}.$$

L is a **context-free language(CFL)** $\Leftrightarrow \exists$ a context-free grammar(CFG) G, such that $L = L(G)$.

Example: Consider the CFG $G = (V, \sum, R, S)$ where $V = \{S, a, b\}, \sum = \{a, b\}, R = \{S \to aSb, S \to e\}$.

$$L(G) = \{a^n b^n : n \geq 0\}$$

$L(G)$ is context-free but not regular

Example: Let $G = (V, \sum, R, S)$ where $V = \{S, (,)\}, \sum = \{(,)\}, R = \{S \to e, S \to SS, S \to (S)\}$.

$L(G)$ is the language containing all strings of balanced parentheses.

Example: Let $G = (\{S, a, b\}, \{a, b\}, R, S)$, where, $R = \{S \to e, S \to SS, S \to aSb, S \to bSa\}$.
$L(G) = \{w \in \{a, b\}^* : w$ has the same number of $a's$ and $b's\}$

Proof:

$w \in L(G) \Rightarrow w$ has the same number of a's and b's.

By Induction on length k of derivation.

(a) $k = 1$

The derivation is $S \Rightarrow e$

$\Rightarrow w = e$ has the same number of a's and b's. ✓

(b) $k > 1$

Then either:

$$S \Rightarrow SS \Rightarrow^* xy = w$$
$$S \Rightarrow aSb \Rightarrow^* axb = w$$
$$S \Rightarrow bSa \Rightarrow^* bxa = w$$

Since $S \Rightarrow^* x, S \Rightarrow^* y$ by derivations of length $< k$
x, y have equal number of a's and b's(IH)

so do xy, axb, and bxa. $\checkmark$

w has the same number of a's and b's $\Rightarrow w \in L(G)$.

By induction on $|w|$.

(a) $|w| = 0$

$$w = e \in L(G)$$

$S \rightarrow e$ is a rule.

(b) $|w| = k + 2$: ($|w|$ must be even)

4 subcase depending on first and last symbols of w:

Case 1 $w = axb$ for some $x \in \sum^*$

$|x| = k$ $\Rightarrow$ $\boxed{S \Rightarrow^* x}$

x has the same number of a's and b's.

$$S \Rightarrow aSb \Rightarrow^* axb = w$$

Case 2 $w = bxa$ for some $x \in \sum^*$ - *similar*

Case 3 $w = axa$ for some $x \in \sum^*$

$|x| = k$, x has 2 more b's than a's.

$x = uv$ for some u and v such that

- u has one more b than a.
- v has one more b than a.

$S \Rightarrow^* au$ and $S \Rightarrow^* va$

So $S \Rightarrow^* SS \Rightarrow^* auva = w$.

Case 4 $w = bxb$ for some $x \in \sum^*$ - *similar*

Example: Consider the CFG $G = (V, \sum, R, S)$ where

– $V = \{+, *, (,), id, T, F, E\}$

– $\sum = \{+, *, (,), id\}$

– $R = \{E \rightarrow E + T, E \rightarrow T, T \rightarrow TF, T \rightarrow F, F \rightarrow (E), F \rightarrow id\}$

E: expression, T: term, F: factor.

Remark:

- Computer programs written in any programming language must be syntactically correct and enable to mechanical interpretation.
- The syntax of most programming language can be captured by CFG.

Example All regular languages are CFL.

Proof: We will encounter several proof of this fact.

- In **section 3.3**:

The CFL are precisely the languages accepted by Push-down Automata, which is a generalization of the FA.

- In **section 3.5**:

The class of CFL is closed under union, concatenation, and Kleene star.

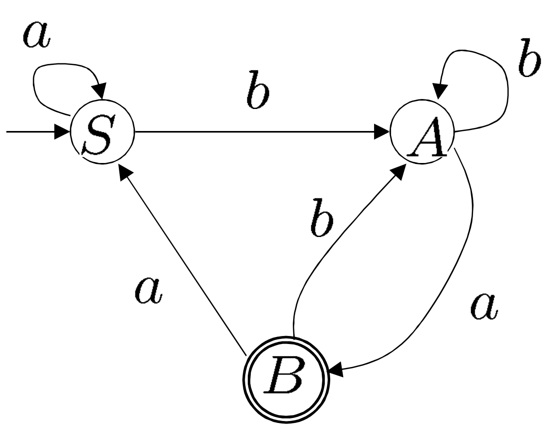
The trivial languages $\emptyset$; and $\{a\}$ are definitely context-free.

• Now we show that all regular languages are CFL by a **direct construction**.

Consider the regular language accepted by the DFA $M = (K, \sum, \delta, s, F)$.

To build a CFG from DFA M

Example:



$\Rightarrow$

$S \rightarrow aS,$
$S \rightarrow bA,$
$A \rightarrow aB,$
$A \rightarrow bA,$
$B \rightarrow aS,$
$B \rightarrow bA,$
$B \rightarrow e.$

Consider string $aabbaba$:

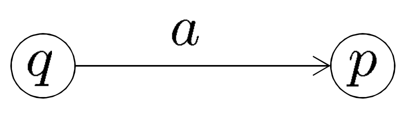
$Saabbaba \vdash_M aSabbaba \vdash_M aaSbbaba \vdash_M aabAbaba \vdash_M aabbAaba \vdash_M aabbaBba \vdash_M aabbabAa \vdash_M aabbabaB$

$S \Rightarrow_G aS \Rightarrow_G aaS \Rightarrow_G aabA \Rightarrow_G aabbA \Rightarrow_G aabbaBba \Rightarrow_G aabbabAa \Rightarrow_G aabbabaA \Rightarrow_G aabbaba$

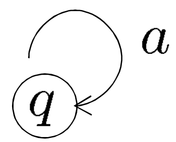
Consider the regular language accepted by the DFA $M = (K, \sum, \delta, s, F)$.

The same language is generated by the CFG $G = (V, \sum, R, S)$ where,

$$
\begin{aligned}
V &= K \cup \sum \\
S &= s \\
R &= \{q \to ap : \delta(q,a) = p\} \\
&\quad \cup \{q \to e : q \in F\}.
\end{aligned}
$$



$q \to ap$



$q \to aq$

3.2 Parse Tree

Example: If G is the CFG that generates language of balanced parentheses.

Then string (())() can be derived from S by at least two distinct derivations, namely,

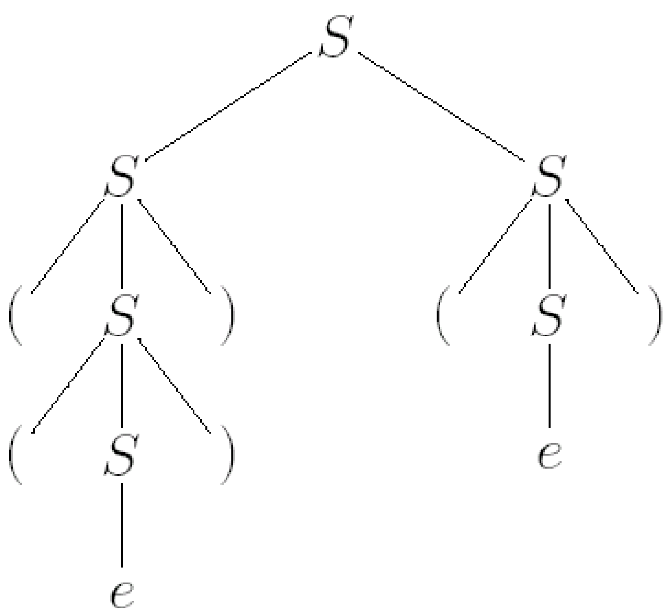
$$\boxed{\begin{array}{l} S \to e \\ S \to SS \\ S \to (S) \end{array}}$$

Same!

$$S \Rightarrow \underline{S}S \Rightarrow (\underline{S})S \Rightarrow ((\underline{S}))S \Rightarrow (())\underline{S} \Rightarrow (())(\underline{S}) \Rightarrow (())()$$

$$S \Rightarrow S\underline{S} \Rightarrow \underline{S}(S) \Rightarrow (S)(\underline{S}) \Rightarrow (\underline{S})() \Rightarrow ((\underline{S}))() \Rightarrow (())()$$

Both derivations can be pictured as following:



- Node: has a label in V
- Root: Start symbol
- Leaves: labeled by terminals
- The string

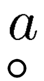
— *by concatenating the labels of leaves from left to right is called the yield of the parse tree.*

— Parse tree

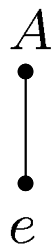
□ **Parse tree:** formal definition

For an arbitrary CFG $G = (V, \sum, R, S)$, we define its parse tree as following:

- This is the parse tree for each $a \in \sum$. The single node is both the root and a leaf. The yield of this parse tree is a.

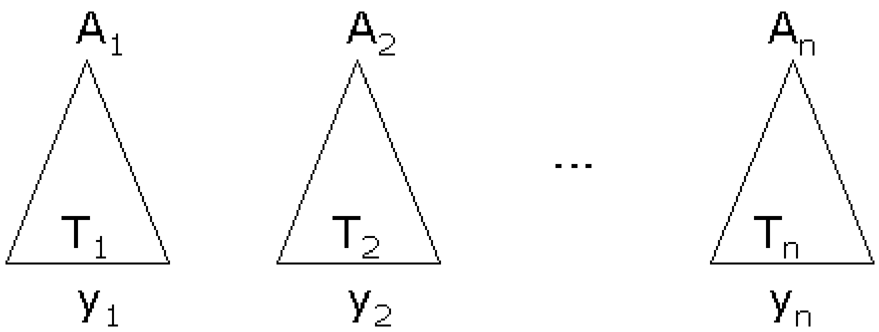


- If $A \rightarrow e$ is a rule in R, then is a parse tree.

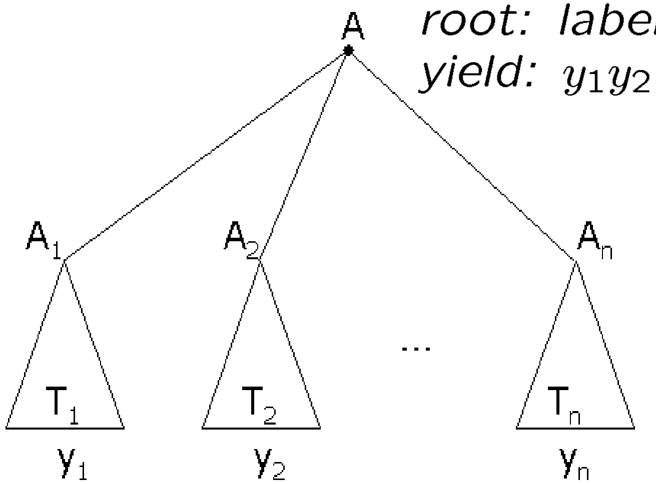


– *root: labeled* A
– *leaf: labeled* e
– *yield:* e

- If



are parse trees, where $n \geq 1$, with roots labeled $A_1, \cdots, A_n$, respectively, and yields $y_1, \cdots, y_n$, and $A \to A_1 \cdots A_n$ is a rule in R, then



root: labeled A
yield: $y_1 y_2 \cdots y_n$.

is a parse tree.

- Nothing else is a parse tree.

Remark:

Parse tree

— Equivalence classes of derivation

□ Derivations D and D' are similar: formal definition

Let $G = (V, \Sigma, R, S)$ be a CFG,

$$D = x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_n$$

$$D' = x'_1 \Rightarrow x'_2 \Rightarrow \cdots \Rightarrow x'_n$$

be two derivations, where $x_i, x'_i \in V^*$ for $i = 1 \cdots n$, $x_1, x'_1 \in V - \Sigma$, and $x_n, x'_n \in \Sigma^*$.

1) D **precedes** $D'(D \prec D') \Leftrightarrow \exists 1 < k < n$, such that

- for all $i \neq k$, we have $x_i = x'_i$.
- $x_{k-1} = x'_{k-1} = uAvBw$, where $u, v, w \in V^*$, and $A, B \in V - \Sigma$.

- $x_k = uyvBw$, where $A \to y \in R$.
- $x'_k = uAvzw$, where $B \to z \in R$.
- $x_{k+1} = x'_{k+1} = uyvzw$.

2) D and D' are **similar** $\Leftrightarrow$ (D, D'), belong in the reflexive, symmetric, transitive closure of $\prec$.

Remark:

- Similarity is an equivalence relation.
- Derivations in the same equivalence class under similarity have the same parse tree.

- Each parse tree contains a derivation that is maximal under $\prec$.

  — leftmost derivation (similarly, rightmost derivation)

Theorem: Let $G = (V, \sum, R, S)$ be a CFG, and let $A \in V - \sum$, and $w \in \sum^*$. Then the following statements are equivalent:

(a) $A \Rightarrow^* w$.

(b) $\exists$ Parse tree with root A and yield w.

(c) $\exists$ a leftmost derivation $A \overset{L}{\Rightarrow}^* w$.

(d) $\exists$ a rightmost derivation $A \overset{R}{\Rightarrow}^* w$.

Example: If G is the CFG that generates language of balanced parentheses. Consider the following derivations D_1, D_2 and D_3:

$$D_1 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow (())S \Rightarrow (())(S) \Rightarrow (())()$$
$$D_2 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow ((S))(S) \Rightarrow (())(S) \Rightarrow (())()$$
$$D_3 = S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow ((S))(S) \Rightarrow ((S))() \Rightarrow (())()$$

$D_1 \prec D_2$, $D_2 \prec D_3$
but not the case $D_1 \prec D_3$

D_1, D_2, and D_3
are similar.

$$D = S \Rightarrow SS \Rightarrow SSS \Rightarrow S(S)S \Rightarrow S((S))S \Rightarrow S(())S \Rightarrow S(())(S) \Rightarrow S(())() \Rightarrow (())()$$

— **not similar to the above**

□ Ambiguity

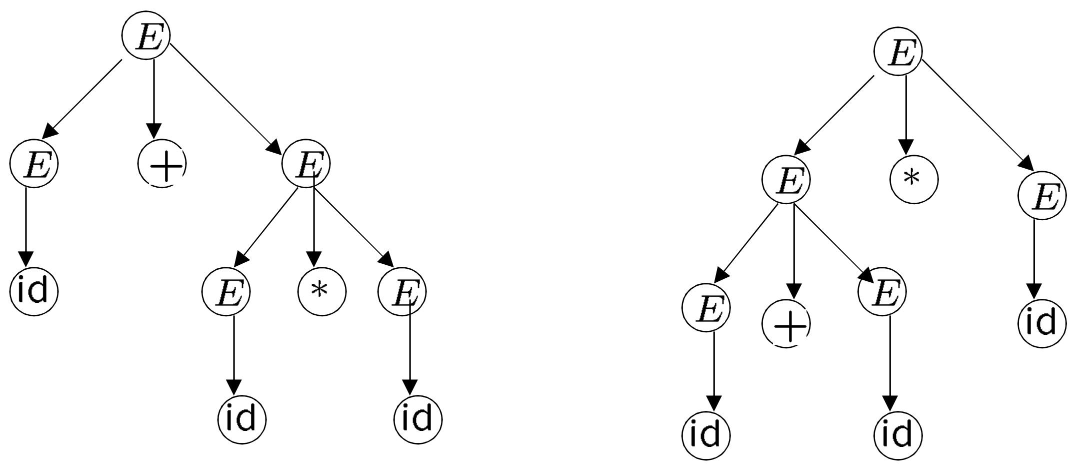
- A leftmost-derivation is a derivation in which a production is always applied to the leftmost symbol.

- In general, a string may have multiple left and rightmost derivations.

Definition: A grammar in which some word has two parse trees is said to be **ambiguous**.

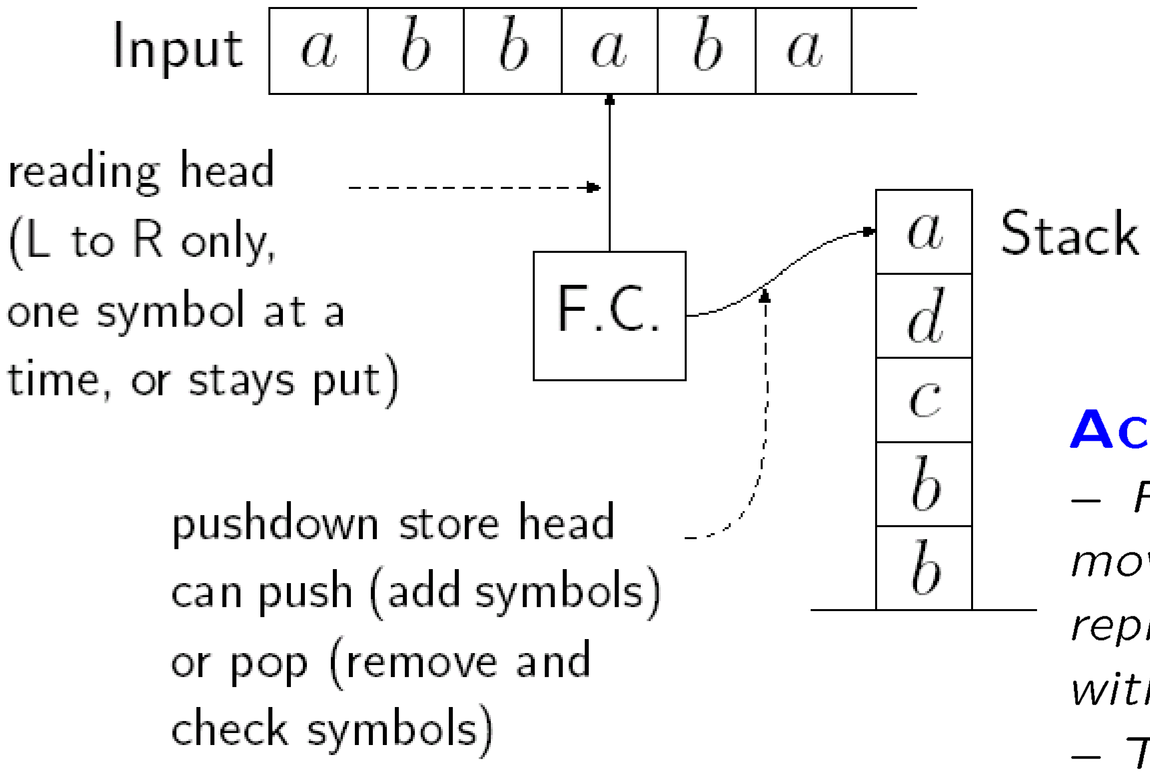
Example: Consider the CFG $G' = (V, \sum, R, S)$ where

- $V = \{+, *, (,), id, E\}$
- $\sum = \{+, *, (,), id\}$
- $R = \{E \rightarrow E + E, E \rightarrow E * E, E \rightarrow (E), E \rightarrow id\}$



Grammar G' is ambiguous.

3.3 Pushdown Automata



PDA has

– *An input tape*
– *A finite control*
– *A stack*

Action:

– *Read: Examine input, move tape head right, and replace the top of the stack with a new symbol*
– *Think: manipulate stack without reading*

Definition: A **pushdown automata(PDA)** is a sextuple $M = (K, \sum, \Gamma, \Delta, s, F)$, where

- K is a finite set of states
- $\sum$ is an alphabet (the input symbols)
- Γ is an alphabet (the stack symbols)
- $s \in K$ is the initial state
- $F \subseteq K$ is the set of final states
- Δ, transition relation, is a subset of

$$(K \times (\sum \cup \{e\}) \times \Gamma^*) \times (K \times \Gamma^*).$$

- **PDA execution:** reading a symbol

Consider $((p, \alpha, \beta), (q, \gamma)) \in \Delta$, Then the PDA can:

  - *enter some state* q
  - *replace* β *by* γ *on the top of the stack*
  - *advance the tape head*

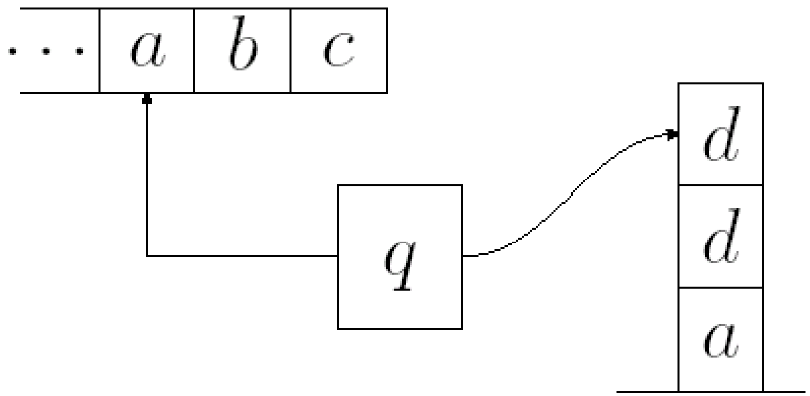
- **PDA execution:** e-transition

Consider $((p, e, \beta), (q, \gamma)) \in \Delta$, Then the PDA can:

  - *enter some state* q
  - *replace* β *by* γ *on the top of the stack*
  - *does not advance the tape head*

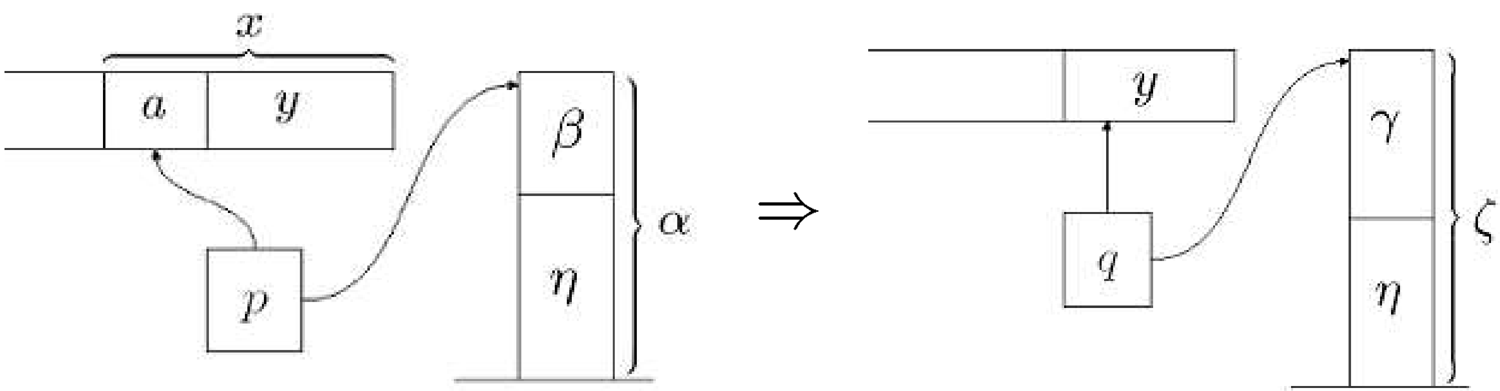
Remark:

- Since several transition of M may be simultaneously applicable at any point, the machines are **nondeterministic**.
- $((p,u,e),(q,a))$— *push* a; $((p,u,a),(q,e))$— *pop* a.
- **Configuration** of a PDA: a member of $K \times \sum^* \times \Gamma^*$.



(q, abc, dda)

- $(p, x, \alpha) \vdash_M (q, y, \zeta)$ (yield in one step) iff there is some transitions $((p, a, \beta), (q, \gamma)) \in \Delta$ such that
  - □ $x = ay$, $a \in \sum \cup \{e\}$
  - □ $\alpha = \beta\eta$
  - □ $\zeta = \gamma\eta$ for some $\eta \in \Gamma^*$



- $\vdash_M^*$ be the reflexive, transitive colsure of $\vdash_M$.

□ Acceptance conditions

A PDA M accepts a string $w \in \sum^*$ iff

- $(s, w, e) \vdash_M^* (p, e, e)$ for some $p \in F$.
- There is a sequence of configuration $C_0, \cdots, C_n (n > 0)$, $(s, w, e) = C_0 \vdash_M C_1 \vdash_M \cdots \vdash_M (p, e, e)$ for some $p \in F$.

Note: Two traditional conditions

– Process the input, and accept if the stack is empty

– Accept if the PDA is in a final state

- The **language accepted by** M:

$L(M) = \{w | (s, w, e) \vdash_M^* (p, e, e)$ for some state $p \in F\}$.

Example: Design a PDA M to accepted the language

$$L = \{wcw^R : \ w \in \{a,b\}^*\}.$$

Solution:

Let $M = (K, \sum, \Gamma, \Delta, s, F)$, where

- $K = \{s, f\}$
- $\sum = \{a, b, c\}$
- $\Gamma = \{a, b\}$
- $F = \{f\}$
- Δ contains five transitions:

$$\boxed{\begin{array}{c} ((s,a,e),(s,a)) \\ ((s,b,e),(s,b)) \\ ((s,c,e),(f,e)) \\ ((f,a,a),(f,e)) \\ ((f,b,b),(f,e)) \end{array}}$$

Example: Design a PDA M to accepted the language

$$L = \{wcw^R : \ w \in \{a,b\}^*\}.$$

State	Unread Input	Stack	Transition Used
s	$abbcbba$	e	–
s	$bbcbba$	a	1
s	$bcbba$	ba	2
s	$cbba$	bba	2
f	bba	bba	3
f	ba	ba	5
f	a	a	5
f	e	e	4

$$((s,a,e),(s,a))$$
$$((s,b,e),(s,b))$$
$$((s,c,e),(f,e))$$
$$((f,a,a),(f,e))$$
$$((f,b,b),(f,e))$$

Example: Design a PDA M to accepted the language

$$L = \{ww^R : \ w \in \{a,b\}^*\}.$$

Solution:

Let $M = (K, \sum, \Gamma, \Delta, s, F)$, where

- $K = \{s, f\}$
- $\sum = \{a, b\}$
- $\Gamma = \{a, b\}$
- $F = \{f\}$
- Δ contains five transitions:

$$\begin{array}{c} ((s,a,e),(s,a)) \\ ((s,b,e),(s,b)) \\ ((s,e,e),(f,e)) \\ ((f,a,a),(f,e)) \\ ((f,b,b),(f,e)) \end{array}$$

Example: Design a PDA M to accepted the language
$L = \{w \in \{a,b\}^* : w$ has the same number of a's and b's$\}$.

Solution:

Let $M = (K, \sum, \Gamma, \Delta, s, F)$, where

- $K = \{s, q, f\}$
- $\sum = \{a, b\}$
- $\Gamma = \{a, b, c\}$
- $F = \{f\}$
- Δ is listed right.

$$
\boxed{\begin{array}{l}
((s,e,e),(q,c)) \\
((q,a,c),(q,ac)) \\
((q,a,a),(q,aa)) \\
((q,a,b),(q,e)) \\
((q,b,c),(q,bc)) \\
((q,b,b),(q,bb)) \\
((q,b,a),(q,e)) \\
((q,e,c),(f,e))
\end{array}}
$$

Example: Design a PDA M to accepted the language $L = \{w \in \{a, b\}^* : w$ has the same number of a's and b's$\}$.

State	Unread Input	Stack	Transition	Comments
s	$abbbabaa$	e	-	Initial configuration.
q	$abbbabaa$	c	1	Bottom marker.
q	$bbbabaa$	ac	2	Start a stack of a's.
q	$bbabaa$	c	7	Remove one a.
q	$babaa$	bc	5	Start a stack of b's.
q	$abaa$	bbc	6	
q	baa	bc	4	
q	aa	bbc	6	
q	a	bc	4	
q	e	c	4	
f	e	e	8	Accepts.

$$((s, e, e), (q, c))$$
$$((q, a, c), (q, ac))$$
$$((q, a, a), (q, aa))$$
$$((q, a, b), (q, e))$$
$$((q, b, c), (q, bc))$$
$$((q, b, b), (q, bb))$$
$$((q, b, a), (q, e))$$
$$((q, e, c), (f, e))$$

Example: Every FA can be trivially viewed a PDA that never operates on its stack.

- Let $M = (K, \sum, \Delta, s, F)$ be a NFA.

- Let $M' = (K, \sum, \emptyset, \Delta', s, F)$ be a PDA.

where $\Delta' = \{((p, u, e), (q, e)) : (p, u, q) \in \Delta\}$

Then, $L(M) = L(M')$.

3.4 Pushdown Automata and Context-Free Language

Languages accepted by
PDA

$\Updownarrow$

Languages generated by
CFG

Context-free Language
CFL

Theorem: The class of languages accepted by PDA is exactly the class of CFL.

I. Building a PDA from a CFG

Lemma: Each Context-Free language is accepted by some PDA.

Proof:

To build the PDA M for CFG $G = (V, \sum, R, S)$ such that

$$L(M) = L(G).$$

Main idea:

Define PDA M to mimics a leftmost derivation of the input string.

□ Construction of PDA

Define PDA $M = (K, \sum, \Gamma, \Delta, s, F)$.

- PDA M has just 2 states.

  $p \sim$ start state

  $q \sim$ final state

- Stack alphabet $\Gamma = V$.

- Let Δ contains the following transitions:

  1) $((p, e, e), (q, S))$

  2) $((q, e, A), (q, x))$ for each rule $A \rightarrow x \in R$

  3) $((q, a, a), (q, e)), \forall a \in \sum$.

Example: Let CFG $G = (V, \sum, R, S)$ with $V = \{S, a, b, c\}$, $\sum = \{a, b, c\}$, and $R = \{S \to aSa, S \to bSb, S \to c\}$, then $L(G) = \{wcw^R : \ w \in \{a, b\}^*\}$.

The corresponding PDA, according to the construction above, is $M = (K, \sum, V, \Delta, s, F)$, with

- $K = \{p, q\}$
- $s = p$
- $F = \{q\}$
- Δ contains the following transitions:

I	$((p, e, e), (q, S))$
II	$((q, e, S), (q, aSa)), ((q, e, S), (q, bSb)), \ ((q, e, S), (q, c))$
III	$((q, a, a), (q, e)), ((q, b, b), (q, e)), ((q, c, c), (q, e))$

- Consider the string $abbcbba$.

Derivation:

$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abbSbba \Rightarrow abbcbba$

Corresponding Computation:

$$
\begin{aligned}
(p, abbcbba, e) &\vdash (q, abbcbba, S) \vdash (q, abbcbba, aSa) \\
&\vdash (q, bbcbba, Sa) \vdash (q, bbcbba, bSba) \\
&\vdash (q, bbcbba, bSba) \vdash (q, bcbba, Sba) \\
&\vdash (q, bcbba, bSbba) \\
&\vdash (q, cbba, Sbba) \\
&\vdash (q, cbba, cbba) \\
&\vdash^* (q, e, e)
\end{aligned}
$$

Zhejiang University/CS/Course/Xiaogang Jin
E-Mail: xiaogangj@cise.zju.edu.cn

□ Verify $L(M) = L(G)$

Claim: Let $w \in \sum^*$ and $\alpha \in (V - \sum)V^* \cup \{e\}$. Then

$$S \overset{L}{\Rightarrow}^{*} w\alpha \iff (q, w, S) \vdash_M^* (q, e, \alpha)$$

The claim will suffice to Lemma. Taking $\alpha = e$ that

$$\begin{tikzcd} S \overset{L}{\Rightarrow}^{*} w \arrow[d] & \iff & (q, w, S) \vdash_M^* (q, e, e) \arrow[d] \\ w \in L(G) & \iff & w \in L(M) \end{tikzcd}$$

□ Proof of Claim

$\Rightarrow$

Suppose that $S \overset{L}{\Rightarrow}^{*} w\alpha$ where $w \in \sum^*$, $\alpha \in (V - \sum)V^* \cup \{e\}$.
Induction on the length of **the leftmost derivation of** w.

Basis step:

If the derivation is of length 0, then $w = e$, and $\alpha = S$, hence indeed $(q, w, S) \vdash_M^* (q, e, \alpha)$.

Induction hypothesis:

Assune that if $S \overset{L}{\Rightarrow}^{*} w\alpha$ by a derivation of length n or less, $n \geq 0$, then $(q, w, S) \vdash_M^* (q, e, \alpha)$.

Induction step:

$$S = u_0 \overset{L}{\Rightarrow} u_1 \overset{L}{\Rightarrow} \cdots \overset{L}{\Rightarrow} u_n \overset{L}{\Rightarrow} u_{n+1} = w\alpha.$$

— be a leftmost derivation of $w\alpha$ from S.

Let $u_n = xA\beta$, A — the leftmost nonterminal of u_n.

$\Rightarrow u_{n+1} = x\gamma\beta$, where $x \in \sum^*, \beta, \gamma \in V^*$, and $A \to \gamma$ is in R.

- $S \overset{L}{\Rightarrow} u_1 \cdots \overset{L}{\Rightarrow} u_n = xA\beta$—a leftmost derivation of length n.

  By the induction hypothesis,

$$(q, x, S) \vdash_M^* (q, e, A\beta) \quad (1)$$

- Since $A \to \gamma$ is a rule in R,

$$(q, e, A\beta) \vdash_M (q, e, \gamma\beta) \quad (2)$$

- Note that $u_{n+1} = w\alpha = x\gamma\beta$.

  $\Rightarrow \exists$ string $y \in \sum^*$ such that $w = xy$ and $y\alpha = \gamma\beta$.

- Rewrite (1) and (2)

$$(q, w, S) \vdash_M^* (q, y, \gamma\beta) \tag{3}$$

- Since $y\alpha = \gamma\beta$

$$(q, y, \gamma\beta) \vdash_M^* (q, e, \alpha) \tag{4}$$

Combining (3) and (4) completes the induction step.

$\Leftarrow$ (Omitted)

II. Building a CFG from a PDA

Lemma: If a language is accepted by a PDA, it is CFL.

Outline of the Proof

- Define the simple PDA
- Convert a PDA to an equivalent simple PDA
- Building a CFG from the simple PDA

□ Definition of Simple PDA

A PDA is **simple** if the following is true:
Whenever $((q, a, \beta), (p, \gamma))$ is a transition of the PDA and q is not the start state, then $\beta \in \Gamma$ and $|\gamma| \leq 2$

In other words, A simple PDA always

− consults its topmost stack symbol, and

− replace it either e, or with single stack symbol, or with two stack symbols.

□ Convert a PDA to an equivalent simple PDA

Let $M = (K, \sum, \Gamma, \Delta, s, F)$ be any PDA,

$\Rightarrow$ construct a simple PDA $M' = (K', \sum, \Gamma \cup \{Z\}, \Delta', s', \{f'\})$ such that $L(M) = L(M')$.

$-s', f' \notin K$ be two new states, $Z \notin \Gamma$ be the stack bottom symbol.

$-K' = K \cup \cdots$

– Δ' contains:

- the transition $((s', e, e), (s, Z))$ (start transition)
- for each $f \in F$, $((f, e, Z), (f', e))$ (final transition)
- all transition of Δ

— replace with equivalent transitions that satisfy the simplicity condition.

Replace transitions with equivalent transitions that satisfy the simplicity condition:

- Get rid of transitions with $\beta \geq 2$.

Consider any transition $((q, a, \beta), (p, \gamma)) \in \Delta'$, where $\beta = B_1 \cdots B_n$, with $n > 1$.

Replace with the following transitions:

$$
\begin{array}{l}
((q, e, B_1), (q_{B_1}, e)) \\
((q_{B_1}, e, B_2), (q_{B_1 B_2}, e)) \\
\quad \vdots \\
((q_{B_1 \cdots B_{n-2}}, e, B_{n-1}), (q_{B_1 \cdots B_{n-1}}, e)) \\
((q_{B_1 \cdots B_{n-1}}, a, B_n), (p, \gamma))
\end{array}
$$

- Get rid of transitions with $\gamma > 2$, without introducing any transitions with $\beta \geq 2$.

Consider any transition $((q, u, \beta), (p, \gamma)) \in \Delta'$, where $\gamma = C_1 \cdots C_m$, with $m \geq 2$.

Replace with the following transitions:

$$((q, u, \beta), (r_1, C_m))$$
$$((r_1, e, e), (r_2, C_{m-1}))$$
$$\vdots$$
$$((r_{m-2}, e, e, (r_{m-1}, C_2))$$
$$((r_{m-1}, e, e), (p, C_1))$$

- Get rid of transitions with $\beta = e$, without introducing any transitions with $\beta \geq 2$ or $\gamma > 2$.

Consider any transition $((q, a, e), (p, \gamma))$ with $q \neq s'$.

Replace any such transitions by all transitions of the form

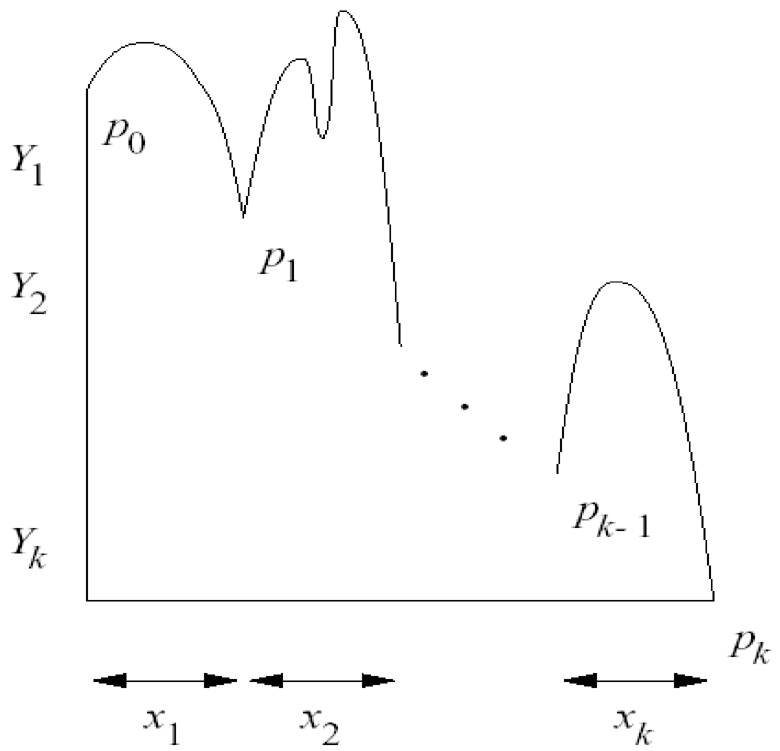
$$((q, a, A), (p, \gamma A) \text{ for all } A \in \Gamma \cup \{Z\}$$

It is easy to see that $L(M) = L(M')$.

□ Construct A CFG from a simple PDA

$\Rightarrow$ Construct a CFG $G = (V, \sum, R, S)$ such that $L(G) = L(M')$.

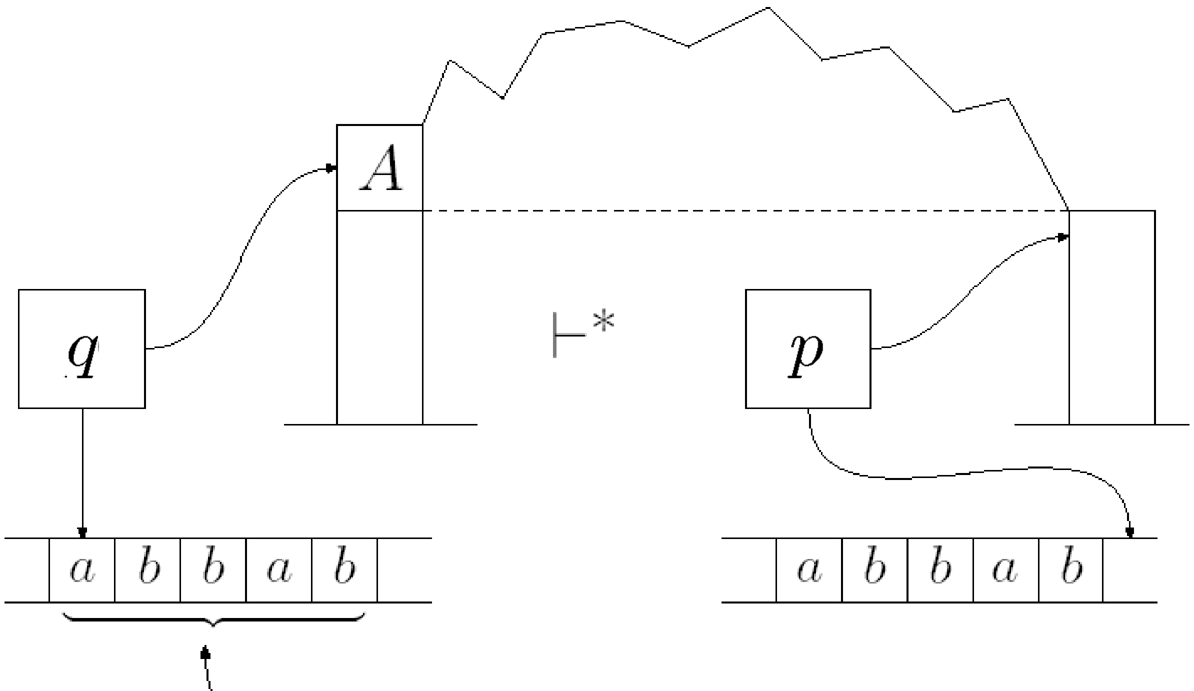
- Let's look at how a PDA can consume $x = x_1x_2\cdots x_k$ and empty the stack.



Definition:

The **nonterminals** $\langle q, A, p\rangle$:
represents any portion of the input string that might be read between a point when M is in state q with A on top of stack, and a point in time when M removes the occurrence of A from the stack and enters state p.

- $V = \{S\} \cup \sum \cup \{\langle q, A, p\rangle | \forall q, p \in K, A \in \Gamma \cup \{e, Z\}\}$



portion of input read from when M is in state q with A on stack to when M enters state p and pops A.

- The rules in R are of four types.

(1) The rules $S \rightarrow \langle s, Z, f' \rangle$:
— s the start state of original PDA and f' the new final state.

(2) For each transition $((q, a, B), (r, e))$, where $q, r \in K$, $a \in \sum \cup \{e\}, B, C \in \Gamma \cup \{e\}$, and for each $p \in K$, we add the rule $\langle q, B, p \rangle \rightarrow a \langle r, C, p \rangle$.

(3) For each transition $((q, a, B), (r, C_1 C_2))$, where $q, r \in K$, $a \in \sum \cup \{e\}, B \in \Gamma \cup \{e\}$, and $C_1, C_2 \in \Gamma$ and for each $p, p' \in K$ we add the rule $\langle q, B, p \rangle \rightarrow a \langle r, C_1, p' \rangle \langle p', C_2, p \rangle$.

(4) For each $q \in K$, the rule $\langle q, e, q \rangle \rightarrow e$.

Example: Let $M = (\{p, q\}, \{0, 1\}, \{X, Z_0\}, \Delta, q, \{p\})$, where Δ contains following transitions:

(1)	$((q, 1, Z_0), (q, XZ_0))$
(2)	$((q, 1, X), (q, XX))$
(3)	$((q, 0, X), (p, X))$
(4)	$((q, e, X), (q, e))$
(5)	$((p, 1, X), (p, e))$
(6)	$((p, 0, Z_0), (q, Z_0))$

Covert PDA M to an equivalent CFG G.

We get CFG $G = (V, \{0, 1\}, R, S)$ where

- $V = \{\langle pXp\rangle, \langle pXq\rangle, \langle pZ_0p\rangle, \langle pZ_0q\rangle, \langle qXp\rangle, \langle qXq\rangle, \langle qZ_0p\rangle, \langle qZ_0q\rangle, S\}$
  - R contains:

$S \to \langle qZ_0q\rangle$

$S \to \langle qZ_0p\rangle$

$$\boxed{((q, 1, Z_0), (q, XZ_0))}$$

From rule (1):

$\langle qZ_0q\rangle \to 1\langle qXq\rangle\langle qZ_0q\rangle$

$\langle qZ_0q\rangle \to 1\langle qXp\rangle\langle pZ_0q\rangle$

$\langle qZ_0p\rangle \to 1\langle qXq\rangle\langle qZ_0p\rangle$

$\langle qZ_0p\rangle \to 1\langle qXp\rangle\langle pZ_0p\rangle$

From rule (2):

$\langle qXq\rangle \rightarrow 1\langle qXq\rangle\langle qXq\rangle$

$\langle qXq\rangle \rightarrow 1\langle qXp\rangle\langle pXq\rangle$

$\langle qXp\rangle \rightarrow 1\langle qXq\rangle\langle qXp\rangle$

$\langle qXp\rangle \rightarrow 1\langle qXp\rangle\langle pXp\rangle$

From rule (4):

$\langle qXq\rangle \rightarrow e$

From rule (5):

$\langle pXp\rangle \rightarrow 1$

From rule (3):

$\langle qXq\rangle \rightarrow 0\langle qXq\rangle$

$\langle qXp\rangle \rightarrow 0\langle pXp\rangle$

From rule (6):

$\langle pZ_0q\rangle \rightarrow 0\langle qZ_0q\rangle$

$\langle pZ_0p\rangle \rightarrow 0\langle qZ_0p\rangle$

Claim: For any $q, p \in K$, $A \in \Gamma \cup \{e\}$, and $x \in \sum^*$,

$$< q, A, p > \Rightarrow_G^* x \Leftrightarrow (q, x, A) \vdash_M^* (p, e, e)$$

The claim will suffice to Lemma.

$$\begin{array}{ccc} < s, e, f > \Rightarrow_G^* x, \text{ for } f \in F & \Longleftrightarrow & (s, x, e) \vdash_M^* (f, e, e) \\ \downarrow & & \downarrow \\ x \in L(G) & \Longleftrightarrow & x \in L(M) \end{array}$$

3.5 Languages that are and are not Context-Free

Closure Properties

Theorem: The CFL are closed under union, concatenation, and Kleene star.

Proof:

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ be two CFG.

Without loss generality, assume that $(V_1 - \Sigma_1)$ and $(V_2 - \Sigma_2)$ are disjoint.

a) Union

Let $G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R, S)$,
where
$R = R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\}$
Then $L(G) = L(G_1) \cup L(G_2)$.

b) Concatenation

Let $G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R, S)$,
where
$R = R_1 \cup R_2 \cup \{S \to S_1 S_2\}$
Then $L(G) = L(G_1)L(G_2)$.

c) Kleene star

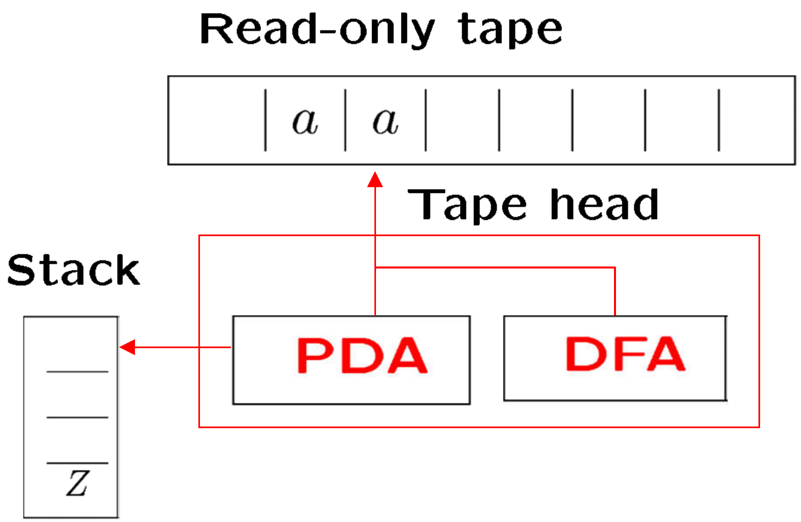
Let $G = (V_1 \cup \{S\}, \sum_1, R, S)$,
where
$R = R_1 \cup \{S \to e, S \to SS_1\}$
Then $L(G) = L(G_1)^*$.

Remark:
CFLs are not closed under intersection and complement.

Theorem: The Intersection of a CFL with a regular language is a CFL.

Proof:

Build a new machine that simulates both automata.



Let $M_1 = (K_1, \sum, \Gamma_1, \Delta_1, s_1, F_1)$ be a PDA and $M_2 = (K_2, \sum, \delta, s_2, F_2)$ be a DFA.

Bulid PDA $M = (K, \sum, \Gamma, \Delta, s, F)$, where

- $K = K_1 \times K_2$
- $\Gamma = \Gamma_1$
- $s = (s_1, s_2)$
- $F = F_1 \times F_2$
- Δ : For each $((q_1, a, \beta), (p_1, \gamma)) \in \Delta_1$, and $q_2 \in K_2$,

$$(((q_1, q_2), a, \beta), ((p_1, \delta(q_2, a)), \gamma)) \in \Delta$$

Example:

$L = \{w : \ w \in \{a, b\}^*, w$ has equal numbers of $a's$ and $b's$
but containing no substring $abaa$ or $babb\}$.

Then L is context-free.

Solution:

$L_1 = \{w : \ w \in \{a, b\}^*, w$ has equal numbers of $a's$ and $b's\}$

L_1 be a CFL
accepted by a PDA

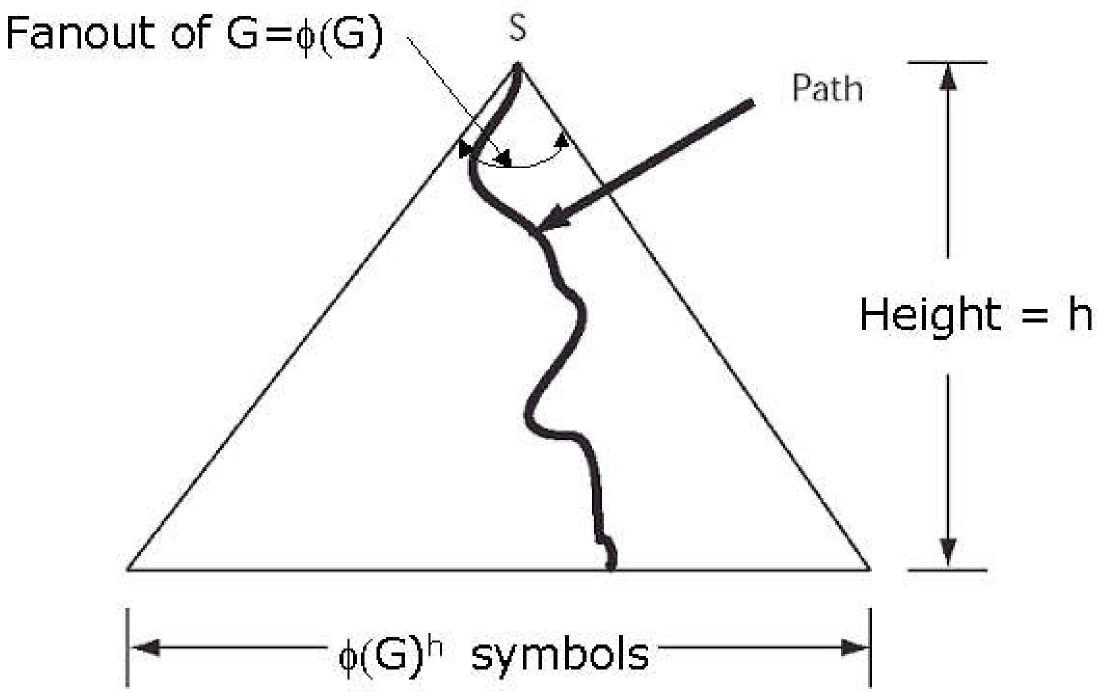
$L_2 = \{w \in \{a, b\}^* : w$ containing no substring $abaa$ or $babb\}$.
$= \{a, b\}^* - \{a, b\}^*(abaa \cup babb)\{a, b\}^*$.

regular

$L = L_1 \cap L_2$ be a context-free language.

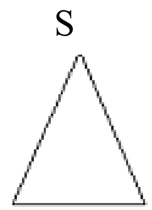
□ Pumping Theorem

Lemma The yield of any parse tree of G of height h has length at most $\phi(G)^h$.

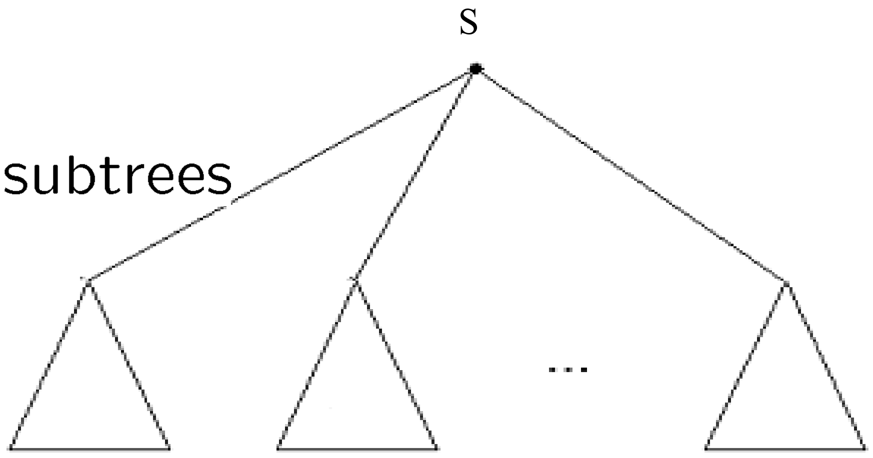


Proof: The proof is by induction on h.

Basis step: $h = 1$



Induction step:



Theorem (Pumping Theorem) Let $G = (V, \sum, R, S)$ be a CFG. Then any string $w \in L(G)$ of length greater than $\phi(G)^{|V-\sum|}$ can be rewritten as $w = uvxyz$ in such way that

- $|vy| \geq 1$
- $uv^n xy^n z \in L(G)$ for every $n \geq 0$.

Proof

- Let w be such a string.
- Let T be the parse tree with root labeled S and with yield w that has the smallest number of leaves.

$|w| > \phi(G)^{|V-\Sigma|}$

$\Rightarrow$ The height of $T > |V - \Sigma|$ (by Lemma)

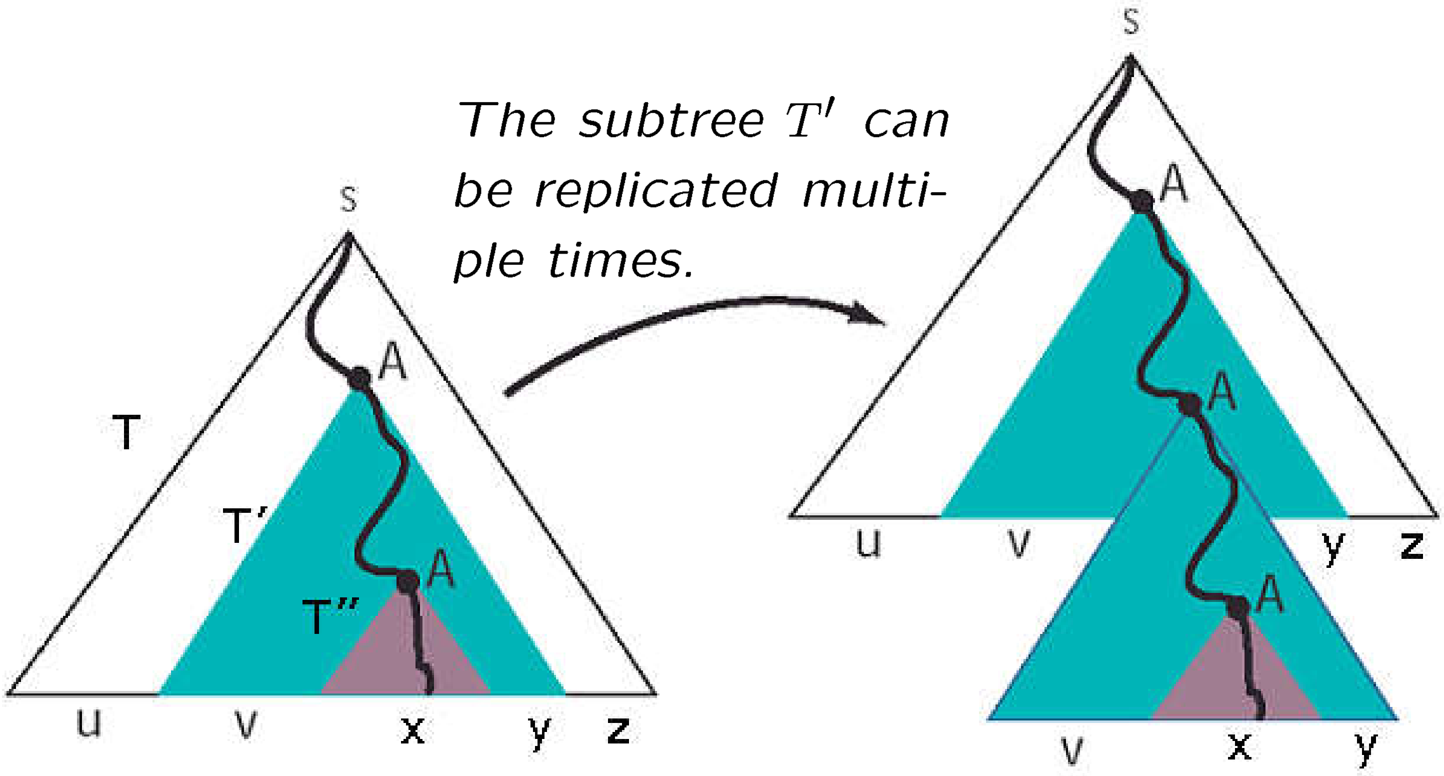
$\Rightarrow$ $\exists$ a path of length at least $|V - \Sigma| + 1$, with at least $|V - \Sigma| + 2$ nodes.

(only one labeled by terminal, the remaining are labeled by nonterminal).

$\Rightarrow$ $\exists$ two nodes on the path labeled with the same symbol.

If $vy = e$, then there is a parse tree with root S and yield w with fewer leaves.
But T is the smallest tree of this kind.
—contradiction!

The subtree T' can be replicated multiple times.



Example:

Show that $L = \{a^n b^n c^n : \ n \geq 0\}$ is not CFL.

Proof: Suppose that $L = L(G)$ for some CFG $G = (V, \sum, R, S)$.

$\longleftarrow$ length $> n$ $\longrightarrow$

a —— a	b —— b	c —— c

Let $n > \frac{\phi(G)^{|V-\sum|}}{3}$.

Then $w = a^n b^n c^n \in L(G)$ and can has a representation $w = uvxyz$ such that

- $vy \neq e$
- $uv^n xy^n z \in L(G)$ for each $n = 0, 1, 2, \cdots$

Question: what about v, y?
Case 1. Contain all three kinds of symbols
Case 2. Contain 2 kinds of symbols

- v, y contains occurrences of all three symbols a, b, c.

$\Rightarrow$ at least one of v, y must contain at least two of them.

$\Rightarrow$ order error in uv^2xy^2z.

- v, y contains occurrences of some but not all of them

$\Rightarrow$ uv^2xy^2z has unequal number of a's, b's and c's.

Contradiction!!

Remark:

Proving that a language is not CFL

- Let L be the proposed CFL

- There is some n, by the pumping lemma

- Choose a string s, longer than n symbols, in the language L

- Using the pumping lemma, construct a new string s that is not in the language

Example: Show $L = \{a^n | n \text{ is prime }\}$ is not Context-free.

Proof: Take a prime $p > \phi(G)^{|V-\sum|}$, where $G = (V, \sum, R, S)$ is CFG and $L = L(G)$.

Then $w = a^p$ can be written as prescribed by Pumping theorem, $w = uvxyz$ and $vy \neq e$.

Suppose that $vy = a^q$ and $uxz = a^r$ where p and q are natural numbers and $q > 0$.

Then the theorem states that $r + nq$ is prime, for all $n \geq 0$.

Contradiction!!

Remark:

Any CFL over a single-letter alphabet is regular.

Example: Show

$L = \{w \in \{a, b, c\}^* | w$ has an equal number of a's, b's and c's$\}$

is not Context-free.

Note:

$$\{a^n b^n c^n | n \geq 0\} = L \cap a^* b^* c^*$$

Theorem: The CFL are not closed under intersection or complementation.

Proof: 1) **Intersection**

Counterexample

$L_1 = \{a^n b^n c^m : m, n \geq 0\}$

$= \{a^n b^n : n \geq 0\} \circ c^*$

$L_2 = \{a^m b^n c^n : m, n \geq 0\}$

L_1 and L_2 are CFL.

$\{a^n b^n c^n : n \geq 0\} = L_1 \cap L_2$ not CFL.

2) **Complementation.**

Note: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Homework 3:	
P120	3.1.3(c) 3.1.9 (a)(b)
P135	3.3.2 (c)(d)
P142	3.4.1
P148	3.5.1 (b)(c)(d) 3.5.2 (c) 3.5.14 (a)(b)(c)(d) 3.5.15