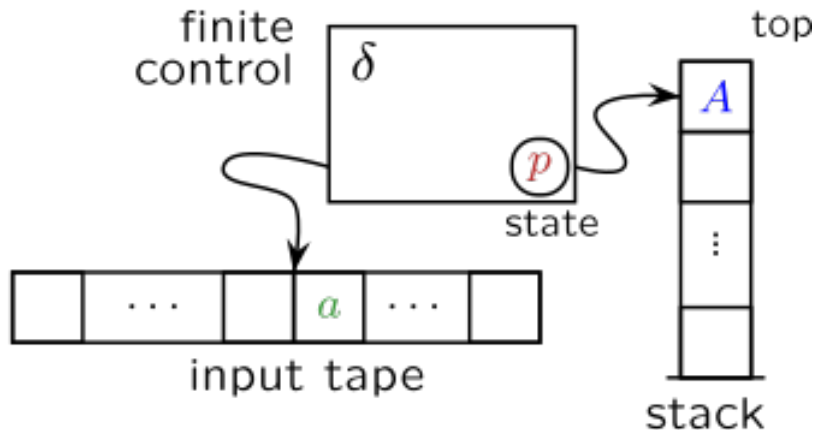


## PDA, the Chomsky hierarchy, and the pumping lemma for context-free languages



Textbook: 2.2 and 2.3

# PushDown Automata (PDA)

- ▶ We have been looking at *finite* state machines
- ▶ What if we need to remember an **infinite** amount information?
- ▶ Would allow us to match the set of strings containing matching parenthesis
- ▶ We can give our state machine an additional memory store in the form of an infinitely large **stack**
- ▶ Allow it to push or pop from that stack
- ▶ **Not** random access: Last in, first out

# nPDA

- ▶ Unlike DFA/NFA, **dPDA** and **nPDA** are not equivalent in **power**
- ▶ Unlike other models, **when we say PDA we mean nPDA not dPDA**
- ▶ We will be ignoring deterministic PDA because they are not equivalent to context-free grammars: nondeterministic PDA are

## Formal definition

**Def:** nPDA. A **nondeterministic pushdown automaton (nPDA)** is a 6-tuple  $(Q, \Sigma, \Gamma, \delta, q_0, F)$  where  $Q, \Sigma, \Gamma, F$  are all finite sets and:

1.  $Q$  is the set of states
2.  $\Sigma$  is the **input alphabet**
3.  $\Gamma$  is the **stack alphabet**
4.  $\delta : (Q \times \Sigma_{\epsilon} \times \Gamma_{\epsilon}) \rightarrow \mathcal{P}(Q \times \Gamma_{\epsilon})$  is the **nondeterministic transition function**
5.  $q_0 \in Q$  is the start state
6.  $F \subseteq Q$  is the set of acceptance states

We let  $\$$  be the special symbol indicating that the stack is empty.

# Interpreting an nPDA transition

For some mapping

$$\delta(q, \sigma, \gamma) = (q', \gamma')$$

- ▶ The current state is  $q$
- ▶ The transition is taken when the input is  $\sigma$  (or  $\epsilon$  for nonconsumptive transitions)
- ▶  $\gamma$  is the item to pop off the stack (or  $\epsilon$  if we don't want to)
- ▶  $q'$  is the new state
- ▶  $\gamma'$  is the item to push to the top of the stack (or  $\epsilon$  if we don't want to)

**Note:**  $\epsilon$  means “don't pop from the stack”,  $\$$  means “empty stack”.  
We disallow the removal of  $\$$ .

# Equivalence with context-free grammars

**Thm:** A language is context free iff some nPDA recognizes it. This proof is 7 pages of the textbook, so we will only go over the broad strokes.

**Lemma 1:** A language being context free implies some nPDA recognizes it. (pg 115)

- ▶ By construction
- ▶ We show how to use an nPDA to determine if the CFG derives any input string

**Lemma 2:** Any language recognized by a nPDA is context free. (pg 119)

- ▶ By construction
- ▶ We show how to create grammar rules from an nPDA in CFG form

# The Chomsky hierarchy

- ▶ The **Chomsky hierarchy** states the power of different models of computation
- ▶ Each entry is a linguistic class and corresponds to a type of automata

In increasing power:

1. Finite-state automata / regular languages
2. Pushdown automata / context-free grammars
3. Linear bounded automata / context-sensitive grammars
4. Turing machines / unrestricted grammars

# Chomsky hierarchy grammar rules

Grammar ⇄	Languages ⇄	Recognizing Automaton ⇄	Production rules (constraints) <sup>[a]</sup> ⇄	Examples <sup>[5][6]</sup> ⇄
Type-3	Regular	Finite-state automaton	$A \rightarrow a$ $A \rightarrow aB$ (right regular) or $A \rightarrow a$ $A \rightarrow Ba$ (left regular)	$L = \{a^n   n > 0\}$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \rightarrow \alpha$	$L = \{a^n b^n   n > 0\}$
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A \beta \rightarrow \alpha \gamma \beta$	$L = \{a^n b^n c^n   n > 0\}$
Type-0	Recursively enumerable	Turing machine	$\gamma \rightarrow \alpha$ ( $\gamma$ non-empty)	$L = \{w   w \text{ describes a terminating Turing machine}\}$

Via Wikipedia



# The pumping lemma for context-free languages

- ▶ Just like we have a pumping lemma to prove that a given language is nonregular, we have another to prove a given language is not context-free
- ▶ Very similar to the pumping lemma for regular languages

**Def:** Pumping lemma for context-free languages. If  $A$  is a context-free language, then there is a number  $p$  (the pumping length) where, if  $s$  is any string  $\in A$  such that  $|s| \geq p$ , then  $s$  may be divided into five pieces  $s = uvxyz$  such that:

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

## CFG Pumping lemma proof

**Pf:** By construction. We will derive a minimal bound for the pumping length  $p$ , then show that any string longer than this must have a derivation where some variable  $R$  derives itself. This implies that the lemma holds, ending the proof.

**Lemma 1:** String sizes. Let  $b$  be the maximum number of variables on the RHS of any rule in our CFG. Then at most  $b$  leaves are within 1 step from the starting variable,  $b^2$  within 2, and  $b^h$  within  $h$ . Thus, any parse tree of height  $h$  produces a string of size at most  $b^h$ . Conversely, if a generated string is at least  $b^h + 1$  long, each of its parse trees must be at least  $h + 1$  high.

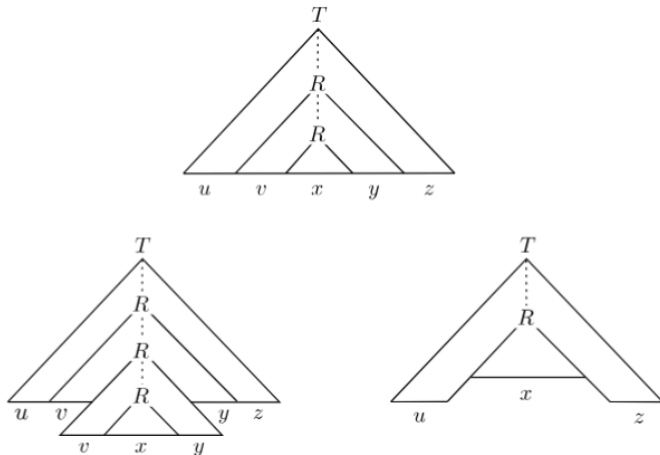
If  $|V|$  is the number of variables in our CFG, we let  $p = b^{|V|+1}$ . Thus, any string longer than the pumping length must have parse trees of height at least  $|V| + 1$ , since  $b^{|V|+1} \geq b^{|V|} + 1$ .

## CFG Pumping lemma proof pt. 2

Since the height of the parse tree is the number of variable replacements, there must be  $|V| + 1$  replacements of  $|V|$  variables. Thus, there must be a variable which is repeated. This variable derives another instance of itself. Thus, we can replace its first occurrence with its final occurrence or infinitely pump the in-between region while remaining in the language.

Conditions 2 and 3 are proven in the book (pg 125).

## CFG Pumping lemma proof pt. 3



**FIGURE 2.35**  
Surgery on parse trees

## Non-context-free languages

**Practice:** Let  $A = \{a^n b^n c^n : n \geq 0\}$ . Prove that  $A$  is not context free.

- ▶ See pg 126 of textbook

Next up: **Part 2: Computability theory**, the Church-Turing thesis, and Turing machines

**An assignment on part 1 of the textbook should come soon**