# Week 11 - PDAs, Pumping Lemma 4 CFLs & WFRs

Billy Price

## PUSHDOWN AUTOMATON

### Definition

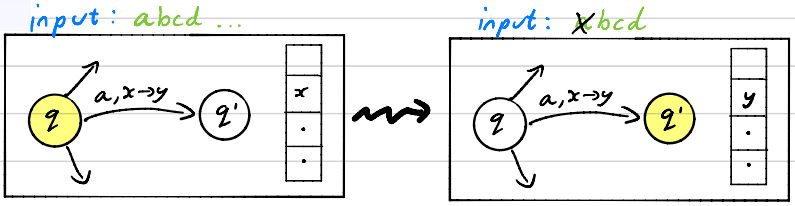
A pushdown automaton is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- $Q$ is a finite set of states,
- $\Sigma$ is the finite input alphabet,
- $\Gamma$ is the finite stack alphabet,
- $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ is the transition function,
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ are the accept states.

### Operation

If
- You are in state $q$
- You have an $x$ on top of the stack
- You can read an $a$ from the input

then you can pick any $(q', y) \in \delta(q, a, x)$, replace the $x$ with $y$ and transition to $q'$




### Epsilon cases

- If $a$ is $\varepsilon$, don't consume any input
- If $x$ is $\varepsilon$, don't take anything off the stack (just add $y$)
- If $y$ is $\varepsilon$, don't replace $x$ with anything (just take off $x$)
- If $x$ is $\varepsilon$ & $y$ is $\varepsilon$, don't change the stack

### PDA Acceptance

The PDA $(Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts input $w$ iff $w = v_1 v_2 \cdots v_n$ with each $v_i \in \Sigma_\epsilon$, and there are states $r_0, r_1, \ldots, r_n \in Q$ and strings $s_0, s_1, \ldots, s_n \in \Gamma^*$ such that

1. $r_0 = q_0$ and $s_0 = \epsilon$.
2. $(r_{i+1}, b) \in \delta(r_i, v_{i+1}, a)$, $s_i = at$, $s_{i+1} = bt$ with $a, b \in \Gamma_\epsilon$ and $t \in \Gamma_\epsilon^*$.
3. $r_n \in F$.

**Note 1:** There is no requirement that $s_n = \epsilon$, so the stack may be non-empty when the machine stops (even when it accepts).

**Note 2:** Trying to pop an empty stack leads to rejection of input, rather than "runtime error".

### Intuitively

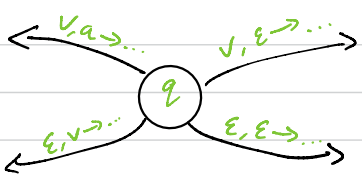
There is some state path you can take from $q_0$ to some accept state which
- consumes all of the input
- always has $x$ on the top of the stack when a transition along the path wants to replace $x$ with something.

**Progressive** A PDA is called progressive iff, for all $q \in Q$, $a \in \Gamma$, $\delta(q, \varepsilon, a) = \phi$

**Deterministic** A PDA is called deterministic iff, from any configuration, there is at most one transition you can take, more precisely...

$$\forall q \in Q \ \forall v \in \Sigma \ \forall a \in \Gamma \left( |\delta(q, v, a) \cup \delta(q, v, \varepsilon) \cup \delta(q, \varepsilon, v) \cup \delta(q, \varepsilon, \varepsilon)| \leq 1 \right)$$

At any state $q$ pick any $v \in \Sigma$, $a \in \Gamma$, then there can be at most one of these four transitions




(we don't care what the "replacement" stack symbols are amongst these transitions)

**Note** The PDAs required for Assignment 2 Challenge 6 are both progressive and deterministic

# PROVING NOT CONTEXT-FREE

Billy Price

**The Pumping Lemma for CFLs** For any language $A$,

$$A \text{ is Context-Free} \Rightarrow \exists p \geq 0 \; \forall s \in A \; \exists u \exists v \exists x \exists y \exists z \left[ \left( |s| \geq p \wedge s = uvxyz \right) \Rightarrow \begin{cases} |vxy| \leq p \wedge \ldots \\ vy \neq \varepsilon \wedge \ldots \\ \forall i \geq 0 \; uv^i x y^i z \in A \end{cases} \right]$$

Equivalently, we can use the contrapositive...

$$\forall p \geq 0 \; \exists s \in A \; \forall u \forall v \forall x \forall y \forall z \left[ |s| \geq p \wedge s = uvxyz \wedge \text{NOT} \begin{cases} |vxy| \leq p \wedge \ldots \\ vy \neq \varepsilon \wedge \ldots \\ \forall i \geq 0 \; uv^i x y^i z \in A \end{cases} \right] \Rightarrow A \text{ is NOT Context Free}$$

① Take any $p$

② design an $s$ at least as long as $p$

③ show any way of breaking down $s$ into $uvxyz$ cannot make all three statements true.

Do ③ by assuming two statements and explaining why the third is not true. Often you will assume $|vxy| \leq p$ & $vy \neq \varepsilon$ and pick a $j \geq 0$ such that $uv^j x y^j z \notin A$

# WELL-FOUNDED RELATIONS

**Definition** Let $X$ be a set and $\prec$ a binary relation over $X$ (this means $\prec \subseteq X \times X$)
We call $\prec$ *well-founded* iff there is no infinite sequence of $X$-elements $x_1, x_2, x_3, \ldots$ such that $x_1 \succ x_2 \succ x_3 \succ \ldots$
In this case, we call $(X, \prec)$ a *well-founded structure*.

This ordering here is useful, usually on something like $\mathbb{N} \times \mathbb{N}$ or $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$

Ordering $X = X_1 \times \cdots \times X_n$ lexicographically:

$$(x_1, \ldots, x_n) \prec (y_1, \ldots, y_n) \text{ iff } \bigvee_{i=1}^{n} \left( x_i \prec_i y_i \wedge \bigwedge_{j=1}^{i-1} x_j = y_j \right)$$

If each $(X_i, \prec_i)$ is well-founded, then so is $(X, \prec)$.

Example using lexicographic ordering: on $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$

$$(2,2,2) \succ (2,1,42) \succ (1,3,1000) \succ (1,3,999) \succ (1,3,0) \succ (0,0,15)$$

**Induction Principle for Well-Founded Structures** Let $(X, \succ)$ be a well-founded structure, and $P(x)$ a property about elements $x \in X$.

*Induction Step* Suppose that $P(y)$ holds for all $y \prec x$, and use this to prove $P(x)$

If the induction step is true for any $x \in X$, then $\forall x \in X \; (P(x))$

**Useful Properties** (Alg State Type ↓ ; well-founded structure ↙)

- $P(x)$: The algorithm, $A$, terminates when starting in state $x$ ↙ $f : Z \to X$
- $P(x)$: The algorithm, $A$, terminates when starting in any state $x$ such that $f(z) = x$

This requires some cleverly defined function $f : Z \to X$, such that if $A$ transitions from state $z$ to state $z'$, then $f(z) \succ f(z')$. This function usually "encodes" the state as a tuple.