

COMP3630/6360: Theory of Computation  
Semester 1, 2022  
The Australian National University

Pushdown Automata

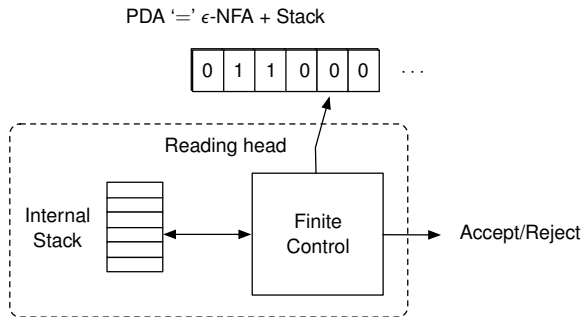
## This lecture covers Chapter 6 of HMU: Pushdown Automata

- Pushdown Automata (PDA)
- Language accepted by a PDA
- Equivalence of CFGs and the languages accepted by PDAs
- Deterministic PDAs

**Additional Reading:** Chapter 6 of HMU.

# Introduction to PDAs

- > PDA '='  $\epsilon$ -NFA + Stack (LIFO)
- > At each instant, the PDA uses:
  - (a) the input symbol, if read; (b) present state; and (c) symbol atop the stack to transition to a new state and alter the top of the stack.
- > Once the string is read, the PDA decides to accept/reject the input string.
- > Note: The PDA can only read a symbol once (i.e., the reading head is unidirectional).

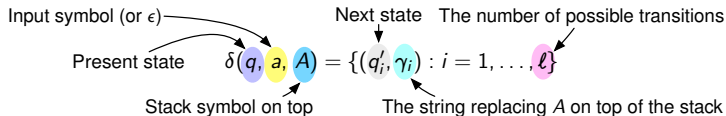


## PDA: Formal Definition

## Definition

A PDA  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  where

- > Just like in DFAs:  $Q$  is the (finite) set of internal states;  $\Sigma$  is the finite alphabet of input tape symbols;  $q_0 \in Q$  is the (unique) start state;  $F$  is the set of final or accepting states of the PDA.
- >  $\Gamma$  is the finite alphabet of stack symbols;
- >  $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$  (power set of  $Q \times \Gamma^*$ ) such that  $\delta(q, a, \gamma)$  is always a finite set of pairs  $(q', \gamma')$   $\in Q \times \Gamma^*$ .
- >  $Z_0 \in \Gamma$  is the sole symbol atop the stack at the start; and

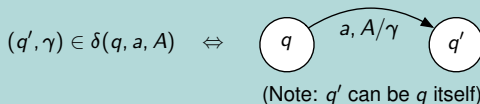


**Convention:** lower case symbols  $s, a,$  and  $b$  will denote input symbols; lower case symbols  $u, v, w$  will exclusively denote strings of input symbols; stack symbols are indicated by upper case letters (e.g.,  $A, B,$  etc); strings of stack symbols are indicated by greek letters (e.g.,  $\alpha, \beta,$  etc);

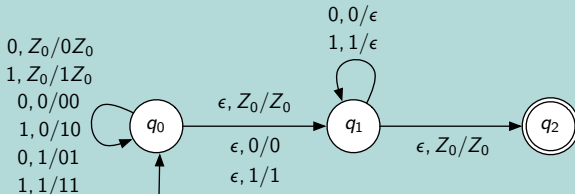
## A PDA Example

## Transition Diagram Notation

Notation: The label  $a, A/\gamma$  on the edge from a state  $q$  to  $q'$  indicates a possible transition from state  $q$  to state  $q'$  by reading the symbol  $a$  when the top of the stack contains the symbol  $A$ . This stack symbol is then replaced by the string  $\gamma$ .



PDA that accepts  $L = \{ww^R : w \in \{0,1\}^*\}$



## Language Accepted by a PDA

## Definitions

- > The **Configuration** or **Instantaneous Description (ID)** of a PDA  $P$  is a triple  $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$  where:
  - (i)  $q$  is the state of the PDA;
  - (ii)  $w$  is the unread part of input string; and
  - (iii)  $\gamma$  is the stack contents from top to bottom.
- > An ID tracks the trajectory/operation of the PDA as it reads the input string.
- > **One-step computation** of a PDA  $P$ , denoted by  $\vdash_P$ , indicates configuration change due to one transition. Suppose  $(q', \gamma) \in \delta(q, a, A)$ . For  $w \in \Sigma^*$ ,  $\alpha \in \Gamma^*$ ,

$$(q, aw, A\alpha) \vdash_P (q', w, \gamma\alpha), \quad [\text{one-step computation}]$$

- > **(multi-step) computation**, denoted by  $\vdash_P^*$ , indicates configuration change due to zero or any finite number of consecutive PDA transitions.
  - >  $ID \vdash_P^* ID'$  if there are  $k$  IDs  $ID_1, \dots, ID_k$  (for some  $k \geq 1$ ) such that:
    - (i)  $ID_1 = ID$  and  $ID_k = ID'$ , and
    - (ii) for each  $i = 1, \dots, k - 1$ ,  $ID_i \vdash_P ID_{i+1}$ .

## Beware of IDs!

## Lemma 6.2.1

Let PDA  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  be given. Let  $q, q' \in Q$ ,  $x, y, w \in \Sigma^*$ , and  $\alpha, \beta, \gamma \in \Sigma^*$ . Then the following hold.

$$(q, x, \alpha) \vdash_P^* (q', y, \beta) \Leftrightarrow (q, xw, \alpha) \vdash_P^* (q', yw, \beta) \quad (1)$$

$$(q, x, \alpha) \vdash_P^* (q', y, \beta) \Rightarrow (q, x, \alpha\gamma) \vdash_P^* (q', y, \beta\gamma) \quad (2)$$

## Proof Idea

- > (1) What hasn't been read cannot affect configuration changes
- > (2) PDA transitions cannot occur on empty stack. So the  $(q, x, \alpha) \vdash_P^* (q', y, \beta)$  must not access any location beneath the last symbol of  $x$ .

Why is the reverse implication of (2) not true?

## Language Accepted by PDAs

## Definition

Given PDA  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , the language accepted by  $P$  by final states is

$$L(P) = \left\{ w \in \Sigma^* : (q_0, w, Z_0) \stackrel{*}{\vdash}_P (q, \epsilon, \alpha) \text{ for some } q \in F, \alpha \in \Gamma^* \right\}.$$

The language accepted by  $P$  by empty(ing its) stack is

$$N(P) = \left\{ w \in \Sigma^* : (q_0, w, Z_0) \stackrel{*}{\vdash}_P (q, \epsilon, \epsilon) \text{ for some } q \in Q \right\}.$$

Can  $L(P)$  and  $N(P)$  be different?

- > Pick a DFA  $A$  such that  $L(A) \neq \emptyset$ . Convert it to a PDA  $P$  by pushing each symbol that is read onto the stack, increasing the stack size each time a symbol is read. For the derived PDA,  $L(P) = L(A)$ . However,  $N(P) = \emptyset$ .
- > Which of the two definitions accepts 'more' languages?

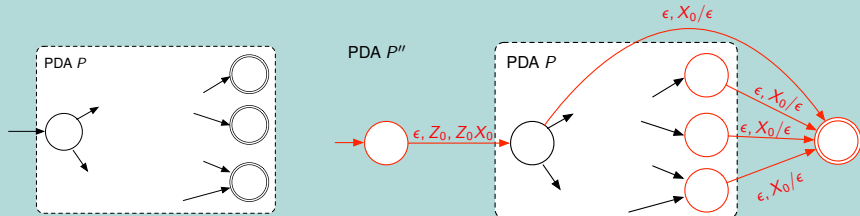


# Equivalence of the Two Notions of Language Acceptance

## Theorem 6.2.2

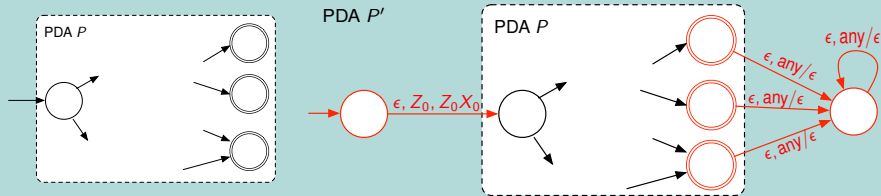
Given PDA  $P$ , there exist PDAs  $P'$  and  $P''$  such that  $L(P) = N(P')$  and  $N(P) = L(P'')$ .

### Proof of Existence of $P''$



- > Introduce a new start state and a new final state with the transitions as indicated.
- > The start state first replaces the stack symbol  $Z_0$  by  $Z_0X_0$ .
- > If and only if  $w \in N(P)$  will the computation by  $P$  end with the stack containing precisely  $X_0$ .
- > The PDA  $P''$  then transitions to the final state popping  $X_0$ . Hence,  $N(P) = L(P'')$ .

## Equivalence of the two Notions of Language Acceptance

Proof of Existence of  $P'$  such that  $L(P) = N(P')$ 

- > Introduce a new start state and a special state with the transitions as indicated.
- > The start state first replaces the stack symbol  $Z_0$  by  $Z_0X_0$ .
- > If and only if  $w \in L(P)$  will the computation by  $P$  end in a final state with the stack containing (at least)  $X_0$ .
- > The PDA  $P'$  then transitions to the special state and starts to pop stack symbols one at a time until the stack is empty. Hence,  $L(P) = N(P')$ .

## CFGs and PDAs

Is every CFL accepted by some PDA and vice versa?

### Theorem 6.3.1

*For every CFG  $G$ , there exists a PDA  $P$  such that  $N(P) = L(G)$ .*

### Proof

- › Let  $G = (V, T, \mathcal{P}, S)$  be given.
- › Construct PDA  $P = (\{q_0\}, T, V \cup T, \delta, S, \{q_0\})$  with  $\delta$  defined by
  - [Type 1]  $\delta(q_0, a, a) = \{(q_0, \epsilon)\}$ , whenever  $a \in \Sigma$ ,
  - [Type 2]  $\delta(q_0, \epsilon, A) = \{(q_0, \alpha) : A \rightarrow \alpha \text{ is a production rule in } \mathcal{P}\}$ .
- › This PDA mimics all possible leftmost derivations.
- › We use induction to show that  $L(G) = N(P)$

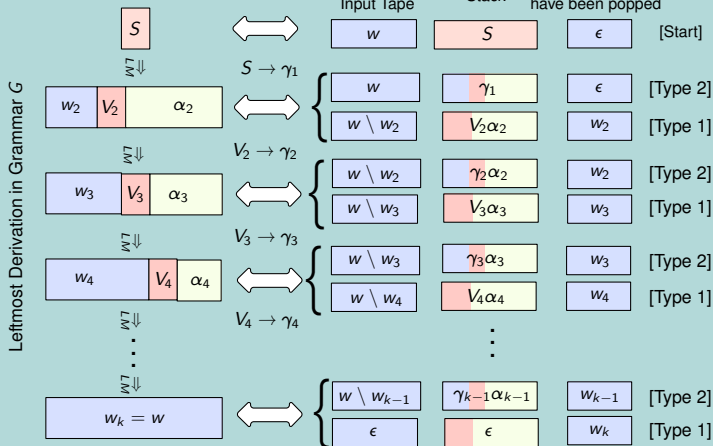
## CFGs and PDAs

## Proof of 1-1 Correspondence between PDA Moves and Leftmost Derivations

Suppose  $w \in T^*$  and  $S \xrightarrow{LM}^* w$ .

$x \setminus y := \text{suffix of } y \text{ in } x$ .

$w_i \in T^*$     $V_i \in V$     $\alpha_i \in (V \cup T)^*$



Configurations in PDA  $P$

$A \setminus B := \text{The suffix of } B \text{ in } A$

## CFGs and PDAs

## Theorem 6.3.2

For every PDA  $P$ , there exists a CFG  $G$  such that  $L(G) = N(P)$ .

## Proof

> Given  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , we define  $G = (V, T, \mathcal{P}, S)$  as follows.

>  $T = \Sigma$ ;

>  $V = \{S\} \cup \{[pXq] : p, q \in Q, X \in \Gamma\}$ ;

Interpretation: Each variable  $[pXq]$  will generate a terminal string  $w$  iff upon reading  $w$  (in finite steps)  $P$  moves from state  $p$  to  $q$  popping  $X$  from the stack.

>  $\mathcal{P}$  contains only the following rules:

>  $S \rightarrow [q_0Z_0p]$  for all  $p \in Q$ .

> Suppose that  $(r, X_1 \cdots X_\ell) \in \delta(q, a, X)$ . Then, for any states  $p_1, \dots, p_\ell \in Q$ ,

$$[qXp_\ell] \rightarrow a[rX_1p_1][p_2X_2p_2] \cdots [p_{\ell-1}X_\ell p_\ell].$$

Note that if  $(r, \epsilon) \in \delta(q, a, X)$ , then  $[qXr] \rightarrow a$ .

> We will show  $[qXp] \xrightarrow{*}_G w \Leftrightarrow (q, w, X) \vdash_P^* (p, \epsilon, \epsilon)$ . The proof is complete by choosing  $q = q_0, X = Z_0$ .

## CFGs and PDAs

Proof of  $(q, w, X) \vdash_P^* (p, \epsilon, \epsilon) \Rightarrow [qXp] \xrightarrow_G^* w$ . (Induction on # of steps of computation)

> Basis: Let  $w \in N(P)$ . Suppose there is a one-step computation  $(q, w, X) \vdash_P (p, \epsilon, \epsilon)$ . Then,  $w \in \Sigma \cup \{\epsilon\}$ . Since  $(p, \epsilon) \in \delta(q, w, X)$ ,  $[qXp] \rightarrow w$  is a production rule.

> Induction: Let  $(q, w, X) \vdash_P^* (p, \epsilon, \epsilon)$ . Let  $a$  be read in the first step of the computation, and let  $w = ax$ . Then the following argument completes the proof.

$$\textcircled{1} \quad (q, w, X) \vdash_P^* (r_1, x, Y_1, \dots, Y_k) \vdash_P^* (p, \epsilon, \epsilon) \xRightarrow{\text{Defn. } G} [qXp] \rightarrow \underbrace{a}_{w=ax} [r_1 Y_1 r_2] [r_2 Y_2 r_3] \cdots [r_k Y_k p] \quad \textcircled{5}$$

$\textcircled{2}$  A portion of  $x$  is read, and  $Y_1$  is popped; more is read,  $Y_2$  is popped, ...

$$\underbrace{(r_1, \overbrace{w_1 w_2 \cdots w_\ell}^{=x}, Y_1 Y_2 \cdots Y_k)} \vdash_P^* (r_2, w_2 \cdots w_k, Y_2 \cdots Y_k) \vdash_P^* (r_3, w_3 \cdots w_k, Y_3 \cdots Y_k) \cdots \vdash_P^* (r_k, w_k, Y_k) \vdash_P^* (p, \epsilon, \epsilon)$$

$$\begin{array}{ccc} \textcircled{3} & (r_1, w_1, Y_1) & (r_2, \epsilon, \epsilon) & (r_2, w_2, Y_2) \vdash_P^* (r_3, \epsilon, \epsilon) & \cdots & (r_k, Y_k p) \xrightarrow_G^* w_k \\ & \downarrow \text{Induc.} & & \downarrow \text{Induc.} & & \downarrow \text{Induc.} \end{array}$$

$$\textcircled{4} \quad [r_1 Y_2 r_2] \xrightarrow_G^* w_1 \quad [r_2 Y_3 r_3] \xrightarrow_G^* w_2$$

## CFGs and PDAs

Proof of  $[qXp] \xRightarrow{*}_G w \Rightarrow (q, w, X) \vdash_P^* (p, \epsilon, \epsilon)$ . (Induction on # of steps of derivation)

> Basis: Let  $[qXp] \xRightarrow{*}_G w$  in one step. Then,  $[qXp] \rightarrow w$  must be a production rule. Consequently,  $(p, \epsilon) \in (q, w, X)$  and  $(q, w, X) \vdash_P^* (p, \epsilon, \epsilon)$ .

> Induction: Let  $[qXp] \xRightarrow{*}_G w$ .

$$\textcircled{4} \quad (r_0, Y_1 \cdots Y_k) \in \delta(q, a, X) \iff (q, a, X) \vdash_P^* (r_0, \epsilon, Y_1 \cdots Y_k)$$

$$\textcircled{1} \quad [qXp] \xRightarrow{LM} a[r_0Y_1r_1][r_1Y_2r_2] \cdots [r_{k-1}Y_kp] \xRightarrow{*}_{LM} w = aw_1 \cdots w_k$$

$$\textcircled{2} \quad \begin{array}{ccc} \begin{array}{c} \Sigma^* \Downarrow^* \\ w_1 \end{array} & \begin{array}{c} \Sigma^* \Downarrow^* \\ w_2 \end{array} & \begin{array}{c} \Sigma^* \Downarrow^* \\ w_k \end{array} \end{array}$$

$$\textcircled{3} \quad \begin{array}{ccc} \text{Induc.} \Downarrow & \text{Induc.} \Downarrow & \text{Induc.} \Downarrow \\ (r_0, w_1, Y_1) \vdash_P^* (r_1, \epsilon, \epsilon) & (r_1, w_2, Y_2) \vdash_P^* (r_2, \epsilon, \epsilon) & (r_{k-1}, w_k, Y_k) \vdash_P^* (p, \epsilon, \epsilon) \end{array}$$

$$\textcircled{5} \quad (q, \underbrace{aw_1w_2 \cdots w_k}_w, X) \vdash_P^* (r_0, w_1 \cdots w_k, Y_1 \cdots Y_k) \vdash_P^* (r_1, w_2 \cdots w_k, Y_2 \cdots Y_k) \vdash_P^* \cdots \vdash_P^* (p, \epsilon, \epsilon)$$

Lemma 6.2.1
Lemma 6.2.1
Lemma 6.2.1

## Deterministic PDAs (DPDAs)

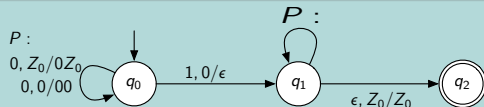
- › PDAs are (by definition) non-deterministic.
- › Deterministic PDAs are defined to have **no choice** in their transitions.

### Definition

A DPDA  $P$  is a PDA  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  such that for each  $q \in Q$  and  $X \in \Gamma$ ,

- ›  $|\delta(q, a, X)| \leq 1$  for any  $a \in \Sigma \cup \{\epsilon\}$ ,  
i.e., a configuration cannot transition to more than one configuration.
- ›  $|\delta(q, a, X)| = 1$  for some  $a \in \Sigma \Rightarrow \delta(q, \epsilon, X) = \emptyset$ ,  
i.e., both reading or not reading (a tape symbol) cannot be options.
- › DPDAs have a computation power that is strictly better than DFAs

Example:  $L(P) = N(P) = \{0^n 1^n : n \geq 1\}$



- › DPDAs have a computation power that is strictly worse than PDAs.  
(We will discuss this later)



## Languages Accepted by DPDAs

- › The two notions of acceptance (empty stack and final state) are **not equivalent** in the case of DPDAs.
- › There are languages  $L$  such that  $L = L(P)$  for some DPDA  $P$ , but there exists no  $P'$  such that  $L = N(P')$ .

### Theorem 6.4.1

*Every regular language  $L$  is the language accepted by the final states of some DPDA.*

### Proof

Simply view the DFA accepting  $L$  as a DPDA (with the stack always containing  $Z_0$ ).

- › The regular language  $L = \{0\}^*$  cannot equal  $N(P)$  for any DPDA  $P$ .
  - › Suppose DPDA  $P$  accepts  $L$  by emptying its stack. Since 0 is accepted,  $P$  eventually reaches a configuration  $(p, \epsilon, \epsilon)$  for some state  $p$ . Now, suppose that  $P$  is fed with the input 00. Since  $P$  is **deterministic**,  $P$  reads a 0 and eventually has to get to  $(p, \epsilon, \epsilon)$ . However, it hangs at this configuration and cannot read any further input symbols. Hence,  $P$  cannot accept 00.

## Languages Accepted by DPDAs

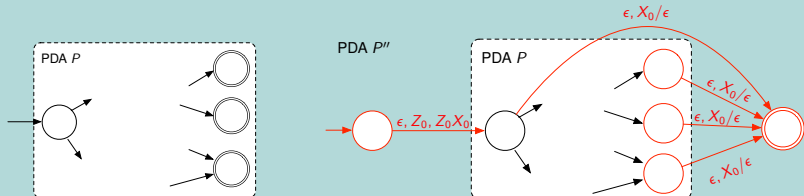
- > A language  $L$  is said to have the **prefix property** if no two distinct strings in the language are prefixes of one another.

### Theorem 6.4.2

A language  $L = N(P)$  for some DPDA  $P$  iff  $L$  has the prefix property and  $L = L(P'')$  for some DPDA  $P''$ .

### Proof $\Rightarrow$

- $\Rightarrow$  Let  $L = N(P)$  for some DPDA  $P$ . Let  $w, ww'$  be in  $L$  with  $w' \neq \epsilon$ . Then  $(q_0, w, Z_0) \vdash_P^* (p, \epsilon, \epsilon)$  for some  $p \in Q$ . The DPDA hangs at this state since the stack is empty. Hence, it cannot accept  $ww'$ . The fact that  $L = L(P'')$  for some DPDA  $P''$  follows from Theorem 6.2.2 since the construction yields a **deterministic** PDA.



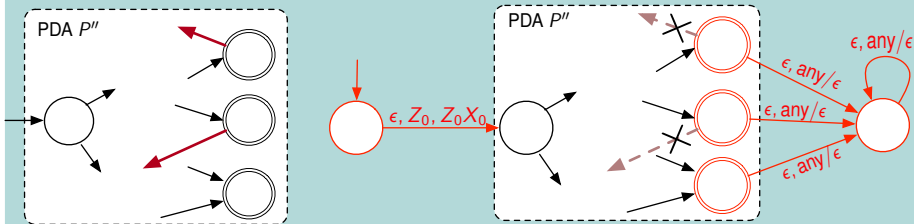
## Languages Accepted by DPDAs

Proof  $\Leftarrow$ 

$\Leftarrow$  Let DPDA  $P''$  be given. Let  $w \in L(P'')$ ,  $(q_0, w, Z_0) \stackrel{*}{\vdash}_P (p, \epsilon, \gamma)$  for some  $p \in F$ , and  $\gamma \in \Gamma$ . Since  $L(P'')$  satisfies the prefix property, the PDA cannot enter any final state before reading all of  $w$ .

> Then we can delete all transitions from final states; this  $X \in \Gamma$  does not alter  $L(P'')$ .

> Then, the construction of Theorem 6.2.2 yields a **deterministic** PDA  $P'$  such that  $N(P') = L(P'') = L$ .



## DPDAs and Unambiguous Grammars

## Theorem 6.4.3

*If  $L = N(P)$  for some DPDA  $P$ , then  $L$  has an unambiguous CFG.*

## Proof

- › Let  $G$  be the CFG constructed in Theorem 6.3.2.
- › Suppose  $G$  is ambiguous. Then, for some  $w \in L$  has 2 leftmost derivations.
- › However, each derivation corresponds to a unique trajectory of configurations in  $P$  that also accepts  $w$  by emptying stack.
- › Since  $P$  is deterministic, the trajectories, and hence, the derivations have to be identical. Hence,  $G$  is unambiguous.

## DPDAs and unambiguous Grammars

## Theorem 6.4.4

*If  $L = L(P)$  for some DPDA  $P$ , then  $L$  has an unambiguous CFG.*

## Proof

- › Let  $\$$  be a symbol not in the alphabet of  $L$ .
- › Consider  $L' = \{w\$ : w \in L\}$ . Then,  $L'$  has the prefix property.
- › By Theorem 6.4.2, there must exist a DPDA  $P'$  such that  $L' = N(P')$ .
- › By Theorem 6.4.3,  $L'$  has an unambiguous CFG  $G' = (V, T, \mathcal{P}, S)$ .
- › Define CFG  $G = (V \cup \{\$\}, T \setminus \{\$\}, \mathcal{P} \cup \{\$ \rightarrow \epsilon\}, S)$ .
- ›  $G$  generates  $L$ .
- › Suppose  $G$  is ambiguous. Then, for some  $w \in L$  has 2 leftmost derivations.
- › The last steps in the two leftmost derivations of  $w$  **must** use the production  $\$ \rightarrow \epsilon$ .
- › Then, the portions of the two leftmost derivations without the last production step correspond to two leftmost derivations of  $w\$$ .
- › Hence,  $G'$  must be unambiguous, which is a contradiction. Hence,  $G$  is also unambiguous.

## Explanation for Slide 11

- >  $\Rightarrow$  Suppose we want to show that if there is a derivation in  $G$  generating  $w$ , then there is a trajectory in  $P$  accepting  $w$ . To do that let  $S \xRightarrow[LM]{*} w$ .
- > Then there must be a LM derivation as in the left column. In each step of the leftmost derivation, a part of the string  $w$  is uncovered, and the uncovered part is succeeded by a non-terminal.
- > Let after  $i = 1, \dots, k - 2$  production uses: (1) the prefix  $w_{i+1}$  of  $w$  be uncovered (shown in purple); (2) the leftmost non-terminal be  $V_{i+1}$  (shown in orange); and (3) is the string to the right of the leftmost non-terminal  $\alpha_{i+1}$  that contains both terminal and non-terminal symbols (shown in beige).
- > After the  $k^{\text{th}}$  production rule, we have derived  $w_k = w$ .
- > Now suppose  $S \rightarrow \gamma_1 = w_2 V_2 \alpha_2$ ,  $V_2 \rightarrow \gamma_2$ ,  $\dots$ ,  $V_{k-1} \rightarrow \gamma_{k-1}$  be the  $k - 1$  production rules used in the leftmost derivation.
- > Now let us show that a trajectory exists for  $P$  using the above information we have laid out.
- > Since there is only one state for the PDA, the right part of the slide presents only the portion of tape yet to be read, and the stack contents; additionally, it also gives the **string of terminals** that has been popped up until any point in time.
- > Initially, the tape contains  $w$ , the stack contains  $S$ , and  $\epsilon$  has been popped thus far.

## Explanation for Slide 11 (Continued)

- Now since  $S \rightarrow \gamma_1$  is a valid production rule, by the definition of  $P$ , there is a Type-2 transition that reads nothing from the input tape, reads  $S$  from the stack and pushes  $\gamma_1 := w_2 V_2 \alpha_2$  onto the stack. Thus, the following one-step computation is valid

$$(q_0, w, S) \vdash_P (q_0, w, w_2 V_2 \alpha_2).$$

- Note that  $w_1$  is the prefix of  $w$  uncovered after the first step of the derivation, and hence matches the first few symbols of  $w$ . Then, it is clear that one can perform  $|w|$  Type-1 transitions that pop each of these symbols from the stack. Thus, after popping  $|w_1|$  symbols, we see that:

$$(q_0, w, S) \vdash_P (q_0, w, w_2 V_2 \alpha_2) \stackrel{*}{\vdash}_P (q_0, w \setminus w_2, V_2 \alpha_2),$$

where we let  $w \setminus w_2$  to denote the suffix of  $w_2$  in  $w$ .

- Now, note that  $V_2 \rightarrow \gamma_2$  is a valid production rule; hence, there is a valid one-step computation from  $(q_0, w \setminus w_2, V_2 \alpha_2)$  that uses the corresponding Type-2 transition. The resultant configuration change will then be

$$(q_0, w, S) \vdash_P (q_0, w, w_2 V_2 \alpha_2) \stackrel{*}{\vdash}_P (q_0, w \setminus w_2, V_2 \alpha_2) \vdash_P (q_0, w \setminus w_2, (w_3 \setminus w_2) V_3 \alpha_3),$$

where  $(w_3 \setminus w_2) V_3 \alpha_3 := \gamma_2 \alpha_2$ .

## Explanation for Slide 11 (Continued)

- › Again, we see that a portion of the top of the stack contains  $w \setminus w_2$ , which matches the initial segment of the input tape. Then there is a valid multi-step computation involving  $|w_3 \setminus w_2|$  Type-1 transitions that pops  $w_3 \setminus w_2$ . The resultant configuration will then be  $q_0, w \setminus w_3, V_3\alpha_3$ .
- › Now, this proceeds until all of  $w$  is exhausted (read) from the input tape, and the configuration at the end will be  $(q_0, \epsilon, \epsilon)$ . Since the stack is empty, the original string  $w$  will be accepted.
- ›  $\Leftarrow$  The direction that a trajectory accepting  $w$  in  $P$  implies a derivation of  $w$  in  $G$  is simply arguing the above in the reverse direction using the facts that:
  - › a trajectory for accepting  $w$  in  $P$  must consist only of Type-1 and Type-2 transitions, and each Type-2 transition corresponds to a unique production in  $G$ .
  - › The argument is literally the same as above except that we now uncover the production rule from the corresponding Type-2 transition.



## Explanation for Slide 13

Inductive proof for  $(q, w, X) \stackrel{*}{\vdash}_P (p, \epsilon, \epsilon) \Rightarrow [qXp] \stackrel{*}{\Rightarrow}_G w$  based on length of computation.

- ▷ Basis: Let  $(q, w, X) \stackrel{*}{\vdash}_P (p, \epsilon, \epsilon)$  be a one-step computation. Thus,  $w$  has to be an input symbol or  $\epsilon$ . Then, by definition of one-step computation it **must** be true that  $(p, \epsilon) \in (q, w, X)$ . Then, by the construction of  $G$ , we have  $[qXr] \rightarrow w$  (see Slide 12 for the construction), and hence  $[qXr] \stackrel{*}{\Rightarrow}_G w$ .
- ▷ Induction:  $(q, w, X) \stackrel{*}{\vdash}_P (p, \epsilon, \epsilon)$  in say  $k > 1$  steps. Let us assume that in the first step of the computation, the symbol  $a$  is read from the input tape (or  $a = \epsilon$ ). Let  $w = ax$ . Let's break the  $k$ -step computation to a single step followed by a  $k - 1$ -step computation as detailed in 1 (encircled in black). Let  $r_1$  be the state of the PDA after the first step and let  $X$  be popped and  $Y_1 \cdots Y_k$  be pushed onto the stack after the first step/transition/move.
- ▷ Now, the claim is that the  $k - 1$  step portion of the computation can be expanded into the sequence of computations as given in 2 (encircled in black). The reasoning is as follows. The ID  $(r_1, x, Y_1 \cdots Y_k)$  eventually changes to  $(p, \epsilon, \epsilon)$ . There must be a finite number of moves after which the effective stack change is the popping of  $Y_1$ , i.e., after a finite number of steps  $Y_2$  is at the top **for the very first time**. The steps until then could have popped  $Y_1$ , pushed a string, and then popped it eventually to reveal  $Y_2$  at the top.

## Explanation for Slide 13 (Continued)

- Let  $w_1$  be the portion of the input tape read and  $r_2$  be the state of the PDA when this intermediate ID where  $Y_2$  is at the top of the stack (i.e., the stack contains  $Y_2 \cdots Y_k$ ) is attained. Thus,

$$(r, x, Y_1 \cdots Y_k) \vdash_P^* (r_2, x \setminus w_1, Y_2, \cdots Y_k) \vdash_P^* (p, \epsilon, \epsilon),$$

where again we let  $w \setminus w_1$  to be the suffix of  $w_1$  in  $w$ .

- By a similar argument, after reading another segment, say  $w_2$ , of the input tape and reaching (some) state  $r_3$ , the top of the stack of the PDA contains  $Y_3$  **for the very first time**. Thus,

$$(r, x, Y_1 \cdots Y_k) \vdash_P^* (r_2, x \setminus w_1, Y_2, \cdots Y_k) \vdash_P^* (r_3, x \setminus (w_1 w_2), Y_3, \cdots Y_k) \vdash_P^* (p, \epsilon, \epsilon).$$

- Proceeding inductively, we see that 2 (encircled in black) holds. Note that  $x$  is then equal to the concatenation of the  $w_i$ 's, i.e.,  $x = w_1 \cdots w_k$ .
- Now focus on the computation within the blue block in 2. In no intermediate ID of the computation is  $Y_2$  at the top of the stack (since  $(r_2, x \setminus w_1, Y_2, \cdots Y_k)$  is the very first time  $Y_2$  is at the top of the stack). Thus, the stack contents  $Y_2 \cdots Y_k$  are never visited in this first set of moves, and hence, we see that

$$(r_1, x, Y_1 \cdots Y_k) \vdash_P^* (r_2, x \setminus w_1, Y_2, \cdots Y_k) \Rightarrow (r_1, w_1, Y_1) \vdash_P^* (r_2, \epsilon, \epsilon). \quad (3)$$

## Explanation for Slide 13 (Continued)

- › Similarly, we see that the in portion of the computation in orange, no intermediate ID of the computation has  $Y_3$  at the top of the stack (since  $(r_3, x \setminus (w_1 w_2), Y_3, \dots Y_k)$  is the very first time  $Y_3$  is at the top of the stack). Hence,

$$(r_2, x \setminus w_2 \cdots w_k, Y_2, \dots Y_k) \stackrel{*}{\vdash}_P (r_3, w_2 \cdots w_k, Y_3 \cdots Y_k) \Rightarrow (r_2, w_2, Y_2) \stackrel{*}{\vdash}_P (r_3, \epsilon, \epsilon). \quad (4)$$

- › We can proceed inductively to argue that  $(r_i, w_i, Y_i) \stackrel{*}{\vdash}_P (r_{i+1}, \epsilon, \epsilon)$  for  $i = 1, \dots, k - 1$ .
- › Now each of these derivations  $(r_i, w_i, Y_i) \stackrel{*}{\vdash}_P (r_{i+1}, \epsilon, \epsilon)$  for  $i = 1, \dots, k - 1$  contain  $k - 1$  or less steps, because the number of steps they contain is at least one-less than the number of steps in the computation in 1 (encircled in black).
- › Consequently, by the induction hypothesis, we have  $[r_i Y_i r_{i+1}] \stackrel{*}{\Rightarrow}_G w_i, i = 1, \dots, k - 1$ .

By the very same argument  $[r_k Y_k p] \stackrel{*}{\Rightarrow}_G w_k$ .

- › Now focus on the yellow box at the top, the first one-step computation guarantees that there exists a production rule

$$[qXp] \rightarrow a[r_1 Y_1 r_2][r_2 Y_2 r_3] \cdots [r_{k-1} Y_{k-1} r_k][r_k Y_k p]. \quad (5)$$

Now combining the above production with the known derivations in 4 (encircled in black), we see that  $[qXp] \stackrel{*}{\Rightarrow}_G aw_1 \cdots w_k = ax = w$ .

## Explanation for Slide 14

Inductive proof for  $(q, w, X) \vdash_P^* (p, \epsilon, \epsilon) \Leftarrow [qXp] \xrightarrow[G]^* w$  based on length of leftmost derivation.

- $\triangleright$  Basis:  $[qXp] \xrightarrow[LM]^* w$  be a one-step derivation. This can be possible only if  $(p, \epsilon) \in (q, w, X)$ , which then means  $(q, w, X) \vdash_P^* (p, \epsilon, \epsilon)$ .
- $\triangleright$  Induction: Let  $[qXp] \xrightarrow[G]^* w$  in  $k > 1$  steps. As in the previous direction, let us split the leftmost derivation into the first step and then rest.
- $\triangleright$  The first step must involve the application of some production rule, say,  $[qXp] \rightarrow a[r_0 Y_1 r_1][r_1 Y_2 r_2] \cdots [r_{k-1} Y_k p]$ .
- $\triangleright$  By 1 (encircled in 1) each non-terminal  $[r_{i-1} Y_i r_i]$   $i = 1, \dots, k$  must derive (via a leftmost derivation) a segment of  $w$ , say  $w_i$  in  $k - 1$  steps or less. [ $w_i$  is the yield of the parse subtree in the parse tree of  $[qXp]$  with yield  $w$ , and the depth of the subtree is at most 1 less than the depth of the parse tree of  $[qXp]$ .].
- $\triangleright$  Hence,  $[r_{i-1} Y_i r_i] \xrightarrow[LM]^* w_i$  for  $i = 1, \dots, k$  in  $k - 1$  steps or less (I've set  $r_k = p$  here).

By induction hypothesis, then  $(r_{i-1}, w_i, Y_i) \vdash_P^* (r_i, \epsilon, \epsilon)$ .
- $\triangleright$  Then by Lemma 6.2.1,  $(r_{i-1}, w_i \cdots w_k, Y_i \cdots Y_k) \vdash_P^* (r_i, w_{i+1} \cdots w_k, Y_{i+1} \cdots Y_k)$ . Thus,

$$(q, w, X) \vdash_P^* (r_0, w_1 \cdots w_k, Y_1 \cdots Y_k) \vdash_P^* (r_1, w_2 \cdots w_k, Y_2 \cdots Y_k) \vdash_P^* (r_k, \epsilon, \epsilon) = (p, \epsilon, \epsilon).$$