Automata, Languages and Computation

Chapter 6 : Push-Down Automata

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Lecture based on material originally developed by :
Gösta Grahne, Concordia University

Push-Down Automata

Introduction

A push-down automaton consists of

- an ϵ-NFA
- a **stack** representing the auxiliary memory

The stack can

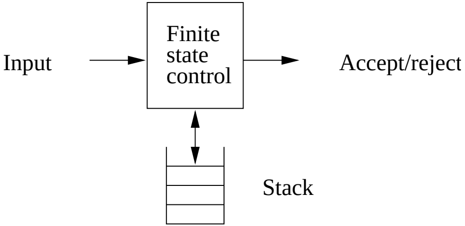
- record an arbitrary number of symbols
- release symbols with a strict policy :
  last in, **first out**

Push-down automata and context-free grammars are equivalent formalisms

Introduction

A transition of a push-down automaton

- consumes a single symbol from the input, or else is an ϵ-transition
- updates the current state
- replaces the **top-most** symbol of the stack stack with a string of symbols, including ϵ



Introduction

More precisely, replacement of symbol X in the stack top-most position with string γ amounts to

- removing X if $\gamma = \epsilon$, also called **pop**
- replacing X if $\gamma = Y$, also called **switch**; if $\gamma = X$, the stack remains unaltered
- inserting new symbols if $|\gamma| > 1$; if $\gamma = ZX$ the transition is called **push**

  First symbol of γ becomes top symbol of the new stack

Example

Let us consider the language (palindrome strings with even length)

$$L_{wwr} = \{ww^R \mid w \in \{0,1\}^*\}$$

generated by the CFG productions

$$P \rightarrow 0P0, \quad P \rightarrow 1P1, \quad P \rightarrow \epsilon$$

Example

A push-down automaton for L_{wwr} has three states, and operates as follows

Guess that you are reading w. Stay in state q_0, and push the input symbol onto the stack

Guess that you are at the boundary between w and w^R. Go to state q_1 using an ϵ-transition

You are now reading the first symbol of w^R. Compare it to the top of the stack. If they match, pop the stack and remain in state q_1. If they don't match, the automaton **halts**, i.e., it does not have a next move

If the stack is empty, go to state q_2 and **accept**

Definition of push-down automaton

A **push-down automaton**, or PDA for short, is a tuple

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

with

- Q finite set of **states**
- Σ finite **input alphabet**
- Γ finite **stack alphabet**
- $\delta : Q \times \Sigma \cup \{\epsilon\} \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ is a **transition** function, always using **finite** subsets of $2^{Q \times \Gamma^*}$
- $q_0 \in Q$ is the initial state
- $Z_0 \in \Gamma$ is the initial stack symbol

  with no symbol in the stack δ is undefined
- $F \subseteq Q$ is the set of final states

Example

The PDA for L_{wwr} is defined as

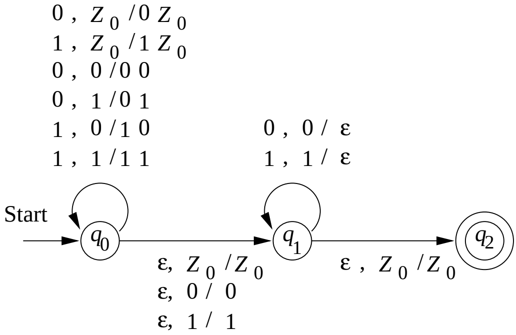
$$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\}),$$

where δ is specified by the following transition table (omitting curley brackets; stack represented as string with top at the left)

	$0, Z_0$	$1, Z_0$	0,0	0,1	1,0	1,1	ϵ, Z_0	$\epsilon, 0$	$\epsilon, 1$
$\rightarrow q_0$	$q_0, 0Z_0$	$q_0, 1Z_0$	$q_0, 00$	$q_0, 01$	$q_0, 10$	$q_0, 11$	q_1, Z_0	$q_1, 0$	$q_1, 1$
q_1			q_1, ϵ			q_1, ϵ	q_2, Z_0		
$\star q_2$									

Example

The transition function δ can also be represented in **graphical** notation, using the convention that $(p, \alpha) \in \delta(q, a, X)$ is associated with an arc from state q to state p with label $a, X/\alpha$



Instantaneous description

Informally, a **computation** of a PDA is a sequence of "configurations" of the automaton obtained one from the other by consuming an input symbol or else by reading ϵ

In order to formalize the configuration of a PDA we introduce the mathematical notion of **instantaneous description**

To formalize the computation of a PDA we then introduce a binary relation over instantaneous descriptions called **moves**

Instantaneous description

An **instantaneous description**, or ID for short, is a triple

$$(q, w, \gamma)$$

where

- q is the current state
- w is the part of the input still to be read
- γ is the stack content, with **topmost symbol** at the left

In this lecture, we will interchangeably use terms instantaneous description and configuration

Computation

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. We define a binary relation over the set of IDs called **moves**, written $\underset{P}{\vdash}$ or also $\vdash$

$\forall w \in \Sigma^*,\ \beta \in \Gamma^*$:

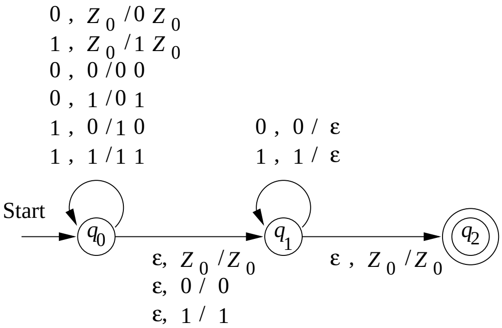
$$(p, \alpha) \in \delta(q, a, X) \Rightarrow (q, aw, X\beta) \vdash (p, w, \alpha\beta)$$
$$(p, \alpha) \in \delta(q, \epsilon, X) \Rightarrow (q, w, X\beta) \vdash (p, w, \alpha\beta)$$

We define $\overset{*}{\underset{P}{\vdash}}$ as the reflexive and transitive closure of $\underset{P}{\vdash}$. We use $\overset{*}{\underset{P}{\vdash}}$ to define a **computation** of a PDA

Compare the above with the two relations rewrite and derivation for a CFG

Example

Given our PDA for L_{wwr}



describe the computation of the automaton for the input 1111

Example

The PDA nondeterministically performs the following computations

$$
\begin{tikzcd}
(q_0, 1111, Z_0) \arrow[d] \arrow[dr] & & \\
(q_0, 111, 1Z_0) \arrow[d] \arrow[dr] & (q_1, 1111, Z_0) \arrow[r] & (q_2, 1111, Z_0) \\
(q_0, 11, 11Z_0) \arrow[d] \arrow[dr] & (q_1, 111, 1Z_0) \arrow[r] & (q_1, 11, Z_0) \arrow[d] \\
(q_0, 1, 111Z_0) \arrow[d] \arrow[dr] & (q_1, 11, 11Z_0) \arrow[dr] & (q_2, 11, Z_0) \\
(q_0, \varepsilon, 1111Z_0) \arrow[d] & (q_1, 1, 111Z_0) \arrow[d] & (q_1, 1, 1Z_0) \arrow[d] \\
(q_1, \varepsilon, 1111Z_0) & (q_1, \varepsilon, 11Z_0) & (q_1, \varepsilon, Z_0) \arrow[d] \\
 & & (q_2, \varepsilon, Z_0)
\end{tikzcd}
$$

Notational conventions for PDAs

We use the following notational conventions

- $a, b, c, \ldots, a_1, a_2, \ldots, a_i, \ldots$ symbols from the input alphabet
- $p, q, r, \ldots, q_1, q_2, \ldots, q_i, \ldots$ states of the automaton
- u, w, x, y, z input strings
- X, Y, Z stack symbols
- $\alpha, \beta, \gamma, \ldots$ stack contents (strings of stack symbols)

Properties of computations

Intuitively, stack or input symbols that are not read/consumed by the PDA **do not affect** the computation :

la stringa non interagisce con la computazione

- if an ID sequence is **valid** (relation $\vdash$), then so is the sequence obtained by adding any string to the tail of the input
- if an ID sequence is valid, then so is the sequence obtained by adding any string to the bottom of the stack lo stack non si svuota mai del tutto
- if an ID sequence is valid and some tail of the input is not consumed, then so is the sequence obtained by removing that tail in every ID in the sequence

Properties of computations

Theorem $\forall w \in \Sigma^*, \gamma \in \Gamma^*$:

y suffisso di x

$$(q, x, \alpha) \overset{*}{\vdash} (p, y, \beta) \Rightarrow (q, xw, \alpha\gamma) \overset{*}{\vdash} (p, yw, \beta\gamma)$$

Note :

- if $\gamma = \epsilon$ we get property 1, and if $w = \epsilon$ we get property 2 from previous slide
- the inverse of the above theorem does not hold: γ can be used in the computation and 'reconstructed' afterward

Theorem $\forall w \in \Sigma^*$:

$$(q, xw, \alpha) \overset{*}{\vdash} (p, yw, \beta) \Rightarrow (q, x, \alpha) \overset{*}{\vdash} (p, y, \beta)$$

Acceptance by final state

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA

The **language accepted by final state** by P is

$$L(P) = \{w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \alpha),\ q \in F\}$$

Note :

- The stack does not necessarily need to be empty at the end of the computation
- The PDA cannot test the end of the string: this is an external condition in the definition of $L(P)$

Example salta questo esempio

Skip this proof. No general technique to prove $L(P) = L$

We show that the PDA P defined in a previous example satisfies $L(P) = L_{wwr}$

(part $\supseteq$) Let $x \in L_{wwr}$. Then $x = ww^R$, and the following is a valid computation

$$\begin{aligned}
(q_0, ww^R, Z_0) &\overset{*}{\vdash} (q_0, w^R, w^R Z_0) \\
&\vdash (q_1, w^R, w^R Z_0) \\
&\overset{*}{\vdash} (q_1, \epsilon, Z_0) \\
&\vdash (q_2, \epsilon, Z_0)
\end{aligned}$$

Example

(part $\subseteq$) Observe that the only way the PDA can enter state q_2 is if it is in state q_1 with the stack containing only Z_0 (empty stack)

Thus it is sufficient to show that if $(q_0, x, Z_0) \overset{*}{\vdash} (q_1, \epsilon, Z_0)$ then $x = ww^R$, for some string w

Using induction on $|x|$, we prove a **more general** property

$$(q_0, x, \alpha) \overset{*}{\vdash} (q_1, \epsilon, \alpha) \quad \Rightarrow \quad x = ww^R$$

Base If $x = \epsilon$ then x is a palindrome

Example

Induction Suppose $x = a_1 a_2 \cdots a_n$, where $n > 0$, and the inductive hypothesis holds for shorter strings

There are two possible moves for P from ID (q_0, x, α)

Move 1 : $(q_0, x, \alpha) \vdash (q_1, x, \alpha)$. Now P can only pop the stack, and any successive computation must have the form

$$(q_1, x, \alpha) \overset{*}{\vdash} (q_1, \epsilon, \beta)$$

with $|\beta| < |\alpha|$

Therefore $\beta \neq \alpha$, and we can never reach the desired ID (q_1, ϵ, α)

Example

Move 2 : $(q_0, a_1 a_2 \cdots a_n, \alpha) \vdash (q_0, a_2 \cdots a_n, a_1\alpha)$. After this move, the only way to reach the desired ID (q_1, ϵ, α) is through a computation with a pop final move

$$(q_1, a_n, a_1\alpha) \vdash (q_1, \epsilon, \alpha)$$

which implies $a_n = a_1$

The intermediate computation must have the form

$$(q_0, a_2 \cdots a_n, a_1\alpha) \overset{*}{\vdash} (q_1, a_n, a_1\alpha)$$

By a previous theorem we can remove symbol a_n. Thus

$$(q_0, a_2 \cdots a_{n-1}, a_1\alpha \overset{*}{\vdash} (q_1, \epsilon, a_1\alpha)$$

By inductive hypothesis, $a_2 \cdots a_{n-1} = yy^R$. Since $a_n = a_1$, $x = a_1 yy^R a_n$ is a palindrome $\square$

Acceptance by empty stack

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be some PDA. The **language accepted by empty stack** by P is

$$N(P) = \{w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \epsilon)\}$$

for any state q

Note : Since final states are no longer relevant in this case, set F is **not used** in the definition ogni tanto la chiede all'esame

From empty stack to final state

Theorem If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, then there exists a PDA P_F such that $L = L(P_F)$

Proof Let

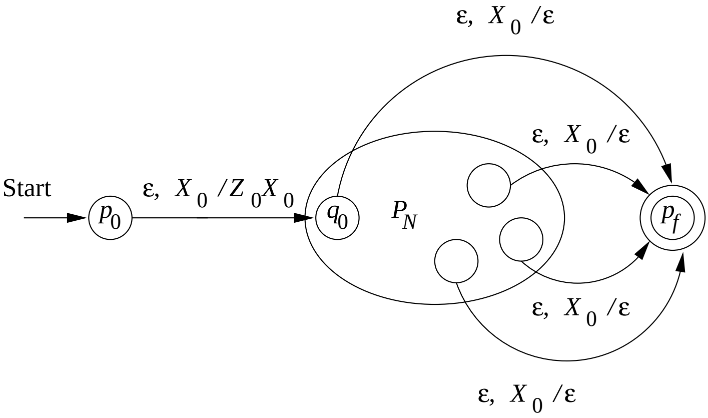
$$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$$

where

- $\delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$
- for each $q \in Q$, $a \in \Sigma \cup \{\epsilon\}$, $Y \in \Gamma$ we let $\delta_F(q, a, Y) = \delta_N(q, a, Y)$
- for each $q \in Q$ we let $(p_f, \epsilon) \in \delta_F(q, \epsilon, X_0)$

From empty stack to final state

Graphical representation of PDA P_F such that $L = L(P_F)$



From empty stack to final state

We need to prove $L(P_F) = N(P_N)$

(part $\supseteq$) Let $w \in N(P_N)$. Then

$$(q_0, w, Z_0) \overset{*}{\underset{N}{\vdash}} (q, \epsilon, \epsilon),$$

for some q. From a previous theorem

$$(q_0, w, Z_0 X_0) \overset{*}{\underset{N}{\vdash}} (q, \epsilon, X_0)$$

Since $\delta_N \subset \delta_F$, we have

$$(q_0, w, Z_0 X_0) \overset{*}{\underset{F}{\vdash}} (q, \epsilon, X_0)$$

From empty stack to final state

We thus conclude

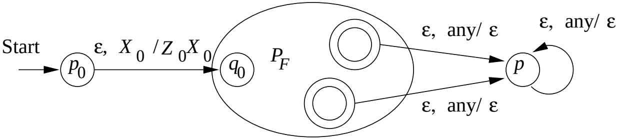
$$(p_0, w, X_0) \underset{F}{\vdash} (q_0, w, Z_0X_0) \overset{*}{\underset{F}{\vdash}} (q, \epsilon, X_0) \underset{F}{\vdash} (p_f, \epsilon, \epsilon)$$

(part $\subseteq$) By inspecting P_F diagram, any accepting computation for w in P_F embeds an accepting computation for w in P_N $\square$

From final state to empty stack

Theorem Let $L = L(P_F)$ for some PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$. There exists a PDA P_N such that $L = N(P_N)$

Construction diagram for P_N from P_F



From final state to empty stack

Proof Let

$$P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$$

where

- $\delta_N(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$
- $\delta_N(q, a, Y) = \delta_F(q, a, Y)$ for each $q \in Q$, $a \in \Sigma \cup \{\epsilon\}$, $Y \in \Gamma$
- $(p, \epsilon) \in \delta_N(q, \epsilon, Y)$, for each $q \in F$, $Y \in \Gamma \cup \{X_0\}$
- $\delta_N(p, \epsilon, Y) = \{(p, \epsilon)\}$, for each $Y \in \Gamma \cup \{X_0\}$

From final state to empty stack

We now prove $N(P_N) = L(P_F)$

(part $\subseteq$) By inspecting P_N diagram, any accepting computation for w in P_N embeds an accepting computation for w in P_F

(part $\supseteq$) Let $w \in L(P_F)$. Then

$$(q_0, w, Z_0) \overset{*}{\underset{F}{\vdash}} (q, \epsilon, \alpha)$$

for some $q \in F, \alpha \in \Gamma^*$

From final state to empty stack

Since $\delta_F \subseteq \delta_N$, and from a previous theorem stating that X_0 can be added to the bottom of the stack, we have

$$(q_0, w, Z_0X_0) \overset{*}{\underset{N}{\vdash}} (q, \epsilon, \alpha X_0)$$

Then P_N can compute

$$(p_0, w, X_0) \underset{N}{\vdash} (q_0, w, Z_0X_0) \overset{*}{\underset{N}{\vdash}} (q, \epsilon, \alpha X_0) \overset{*}{\underset{N}{\vdash}} (p, \epsilon, \epsilon)$$

$\square$

Exercises

Specify a PDA accepting by **final state** the language

$$L = \{a^n b^n c^i \mid n \geqslant 1,\ i \geqslant 1\}$$

and informally explain the way computations work

Specify a PDA accepting by **empty stack** the language

$$L = \{c^i a^n b^n \mid n \geqslant 1,\ i \geqslant 1\}$$

and informally explain the way computations work

Exercises

Specify a PDA accepting by **empty stack** the language

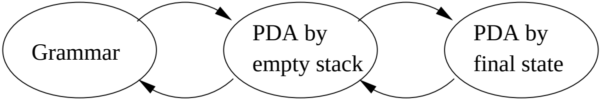
$$L = \{w \in \{0,1,2\}^+ \mid w = x2x',\ x, x' \in (0+1)^*,\ x' = x^R\}$$

and informally explain the way computations work

Equivalence of PDAs and CFGs

Let L be a language. The following statements are equivalent

- L is generated by a CFG
- L is accepted by a PDA by empty stack
- L is accepted by a PDA by final state



We have already seen the equivalence between empty stack and final state

From CFG to PDA

SOLO IDEA generale dellla dimostrazione

Given G, we specify a PDA P_G accepting by empty stack and simulating the relation $\overset{*}{\underset{lm}{\Rightarrow}}$

We write left sentential forms as $xA\alpha$, where A is the leftmost variable and $A\alpha$ is called the **tail** of the form

Example :

$$\underbrace{(a+}_{x}\underbrace{\underbrace{E}_{A}\underbrace{)}_{\alpha}}_{\text{tail}}$$

From CFG to PDA

P_G makes use of only one state q, therefore no relevant information is encoded into states of the PDA

Let $w = xy$. The leftmost sentential form $xA\alpha$ is represented by the ID $(q, y, A\alpha)$ of P_G that

- has consumed input x
- has input y still to be processed
- has tail $A\alpha$ on the stack

From CFG to PDA

A derivation step

$$xA\alpha \underset{lm}{\Rightarrow} x\beta\alpha$$

is simulated by P_G with a **nondeterministic** move from ID $(q, y, A\alpha)$ to ID $(q, y, \beta\alpha)$

In the ID $(q, ay, a\alpha)$, P_G moves **deterministically** to ID (q, y, α), removing a from both the stack and the input

In all remaining cases, the PDA halts in an **error** condition

From CFG to PDA

Formally, let $G = (V, T, R, S)$ be some CFG. We define P_G as

$$(\{q\}, T, V \cup T, \delta, q, S),$$

where

- $\delta(q, \epsilon, A) = \{(q, \beta) \mid (A \to \beta) \in R\}$ for each $A \in V$
- $\delta(q, a, a) = \{(q, \epsilon)\}$ for each $a \in T$

If all the nondeterministic choices are **correct**, P_G completes the processing of the input with an empty stack

Example

Consider the CFG for arithmetic expressions

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
E & \rightarrow & I \mid E * E \mid E + E \mid (E)
\end{array}
$$

The transition function of the PDA is

$$
\begin{array}{rcl}
\delta(q, \epsilon, I) & = & \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\} \\
\delta(q, \epsilon, E) & = & \{(q, I), (q, E * E), (q, E + E), (q, (E))\} \\
\delta(q, X, X) & = & \{(q, \epsilon)\}, \ \ \forall X \in \{a, b, 0, 1, (,), +, *\}
\end{array}
$$

From CFG to PDA

Theorem $N(P_G) = L(G)$

Proof (Part $\supseteq$) Let $w \in L(G)$. Then we can write

$$S = \gamma_1 \underset{lm}{\Rightarrow} \gamma_2 \underset{lm}{\Rightarrow} \cdots \underset{lm}{\Rightarrow} \gamma_n = w$$

Let $\gamma_i = x_i \alpha_i$ and let $w = x_i y_i$. We show by induction on i that if $S \overset{*}{\underset{lm}{\Rightarrow}} \gamma_i$ then $(q, w, S) \overset{*}{\vdash} (q, y_i, \alpha_i)$

From CFG to PDA

Base $i = 1$. Then $\gamma_1 = S$, $x_1 = \epsilon$ and $y_1 = w$. Therefore $(q, w, S) \overset{*}{\vdash} (q, w, S)$

Induction By the inductive hypothesis $(q, w, S) \overset{*}{\vdash} (q, y_i, \alpha_i)$. We have to show that $(q, y_i, \alpha_i) \overset{*}{\vdash} (q, y_{i+1}, \alpha_{i+1})$

From our hypotheses, α_i begins with a variable and we can write

$$\underbrace{x_i A \chi}_{\gamma_i} \underset{lm}{\Rightarrow} \underbrace{x_{i+1} \beta \chi}_{\gamma_{i+1}}$$

From CFG to PDA

From the inductive hypothesis, $A\chi$ is in the stack, and y_i is the remaining portion of the input. According to P_G definition, we can make the move

$$(q, y_i, A\chi) \vdash (q, y_i, \beta\chi)$$

using a transition of the **first** type

Let us write $\beta\chi = u\beta'$, where u is the longest prefix (including ϵ) of $\beta\chi$ that is entirely composed of terminal symbols. We can now remove the terminal symbols of u from the stack, and eliminate the corresponding terminal symbols y_i, using transitions of the **second** type

From CFG to PDA

In this way we reach the ID $(q, y_{i+1}, \alpha_{i+1})$, with $\alpha_{i+1} = \beta'$ representing the tail of the leftmost sentential form $x_i u \beta' = \gamma_{i+1}$

Finally, since $\gamma_n = w$, we have $\alpha_n = \epsilon$ e $y_n = \epsilon$, and thus $(q, w, S) \overset{*}{\vdash} (q, \epsilon, \epsilon)$. Therefore $w \in N(P_G)$

From CFG to PDA

(Part $\subseteq$) We prove the more general statement :

$$\text{if } (q, x, A) \overset{*}{\vdash} (q, \epsilon, \epsilon), \text{ then } A \overset{*}{\Rightarrow} x$$

In words, if P_G makes a computation that

- consumes an input string x
- removes a variable A from the top of the stack
- does not read/consume the portion of the stack below A

then, in the CFG G, nonterminal A generates x

We prove the statement above by induction on the length of the computation of P_G

From CFG to PDA

Base Computation length 1. Then $A \rightarrow \epsilon$ must be a production of G, $x = \epsilon$, and P_G makes a transition of the **first** type. Therefore $A \Rightarrow \epsilon$

Induction Computation length $n > 1$: the inductive hypothesis holds for any computation having length smaller than n

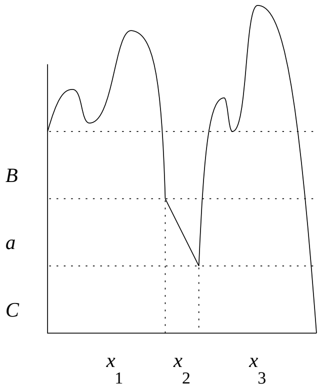
Since A is a variable, the computation must start with a transition of the **first** type

$$(q, x, A) \vdash (q, x, Y_1 Y_2 \cdots Y_k) \vdash \cdots \vdash (q, \epsilon, \epsilon)$$

where $A \rightarrow Y_1 Y_2 \cdots Y_k$ is a production of G

From CFG to PDA

We factorize x in $x = x_1 x_2 \cdots x_k$, as in the following example where $k = 3$, $Y_1 = B$, $Y_2 = a$, e $Y_3 = C$



From CFG to PDA

We obtain that, for every $i \in \{1, \ldots, k\}$, the computation

$$(q, x_i x_{i+1} \cdots x_k, Y_i) \overset{*}{\vdash} (q, x_{i+1} \cdots x_k, \epsilon)$$

has fewer than k steps

If Y_i is a variable, we use the inductive hypothesis to write

$$Y_i \overset{*}{\Rightarrow} x_i$$

If Y_i is a terminal symbol, then $|x_i| = 1$ and $Y_i = x_i$. Therefore $Y_i \overset{*}{\Rightarrow} x_i$ from the reflexive property of $\overset{*}{\Rightarrow}$

From CFG to PDA

We can now compose the desired derivation

$$
\begin{aligned}
A &\Rightarrow Y_1 Y_2 \cdots Y_k \\
&\stackrel{*}{\Rightarrow} x_1 Y_2 \cdots Y_k \\
&\vdots \\
&\stackrel{*}{\Rightarrow} x_1 x_2 \cdots x_k = x
\end{aligned}
$$

From CFG to PDA

To derive the statement of the theorem, we let $A = S$ e $x = w$

Assume $w \in N(P_G)$. Then $(q, w, S) \overset{*}{\vdash} (q, \epsilon, \epsilon)$, and using the general property above we have $S \overset{*}{\Rightarrow} w$, and thus $w \in L(G)$ $\square$