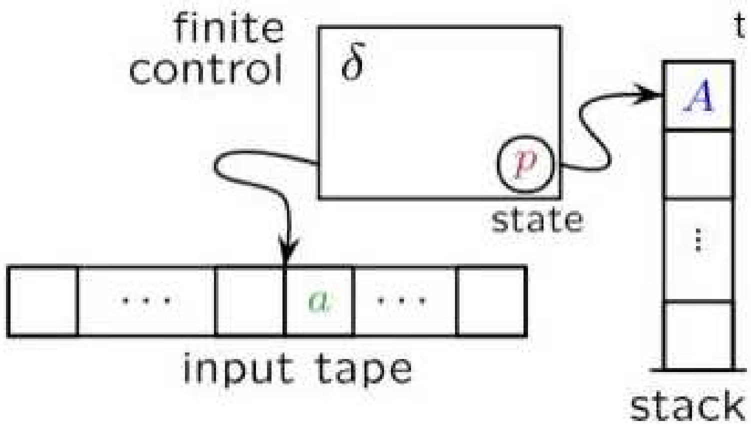
Procedural descriptors for CFLs

- In our study of regular languages we saw that we can describe such languages “procedurally” using a finite automaton (DFA, NFA, e-NFA), or “declaratively” using a regular expression.
- In our study of context free languages, we first learned how we can use CFGs to “declaratively” describe those languages.
- Today we will learn how we can specify machines called Pushdown Automata that describe CFLs “procedurally”.

Pushdown Automata

- A pushdown automata (PDA) is essentially an ϵ-NFA with a stack on which it can store symbols.
- PDA can only access information on the stack in a last-in-first out way.
- There are two versions of a PDA, and they differ on how a PDA accepts a string, i.e.,
  - acceptance by accepting state, or
  - acceptance by empty stack.

PDA informally



- The finite control reads one symbol at a time from the input, and observes the symbol on top of the stack.
- It bases its transitions on its state, the input symbol, and the symbol on top of the stack.
- On a transition, the PDA:
  1. Consumes an input symbol.
  2. Goes to a new state (or stays in the old).
  3. Replaces the top of the stack by a string

An informal example

- Consider the grammar $P \to aPb \mid \epsilon$, and its language

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

- Suppose you are a simple machine whose only memory is a stack.
- How would you recognize strings in this language?

An informal example

Consider the grammar $P \to aPb \mid \epsilon$, and its language

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

A PDA for L has 3 states and operates as follows:

1. Keeps reading as and pushing them onto the stack until it finds a b, which makes it pop an a from the stack and transition to the next state.
2. Keeps reading bs and popping as many as; if the "start symbol" is on top of the stack, then accepts.

PDA, formally

A PDA is a 7-tuple $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where:

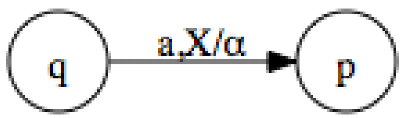
- $Q, \Sigma, q_0,$ and F, are our old friends;
- Γ is the stack alphabet;
- Z_0 is the start symbol;
- δ is the transition function $\delta(q, a, X) = \{(p, \gamma), \cdots\}$, where:
  - q is a state, a is an input symbol, X is a stack symbol
  - p is a state, and γ is the string of stack symbols that replaces X on top of the stack. E.g.:

$\delta(q, a, X) = \{(p, \epsilon)\}$ Stack is popped.
$\delta(q, a, X) = \{(p, X)\}$ Stack is unchanged.
$\delta(q, a, X) = \{(p, YZ)\}$ X replaced by Z; Y pushed.

A Graphical notation for PDAs

In the notation used in our textbook, a transition diagram for a PDA consists of the following elements:

- Like in finite automata, nodes represent states, and the start state and accept states are denoted as usual.
- Arcs correspond to transitions of the PDA as follows:



means $(p, \alpha) \in \delta(q, a, X)$

- Conventionally, Z_0 or Z denote the start symbol for the stack.

Let's obtain a PDA for the language $\{a^n b^n, n \geq 0\}$ using JFLAP.

A Nondeterministic PDA

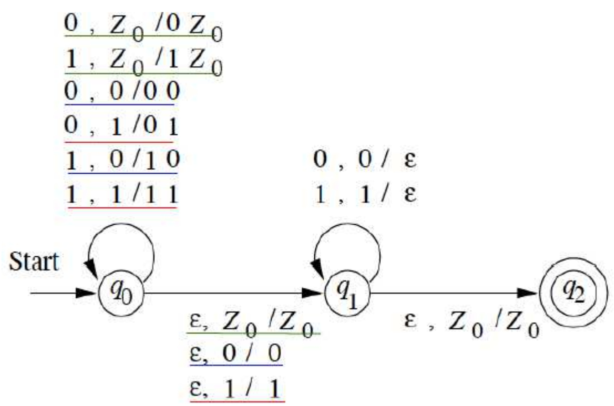
Consider the grammar $P \to 0P0 \mid 1P1 \mid \epsilon$, and its language

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

A PDA for L has three states and operates as follows:

1. Guesses that it is reading w. Stays in state q_0, and pushes the input symbol onto the stack.
2. Guesses that it is in the middle of ww^R, *i.e.*, w would be on the stack. Goes spontaneously to state q_1.
3. It is now reading the head of w^R. Compares it to the top of the stack. If they match, pops the stack, and remains in state q_1. If they don't match, goes to sleep.
4. If the stack is empty, goes to state q_2 and accepts.

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



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

where δ is given by the table below

	$0, Z_0$	$1, Z_0$	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		
$\star q_2$									

We assume the PDA accepts a string by consuming it and entering an accepting state.

PDA: exercise

Design a PDA that recognizes the language

$$\{a^i b^j c^k : i, j, k \geq 1 \text{ and } i = j \text{ or } i = k\}$$

Provide a CFG for the language.

PDA: exercise

Design a PDA that recognizes the language

$$\{w : w \text{ contains as many 1s as 0s}\}$$

Provide a CFG for the language.

PDA: exercise

Design a PDA that recognizes the language

$$\{w : w \text{ contains more 1s than 0s}\}$$

Provide a CFG for the language.

Instantaneous Descriptions (IDs)

- (q, w, γ) is an Instantaneous Description of a PDA configuration.
  - q is the current state
  - w is the remaining input
  - γ is the contents the stack
- Using IDs to represent a computation step by a PDA:

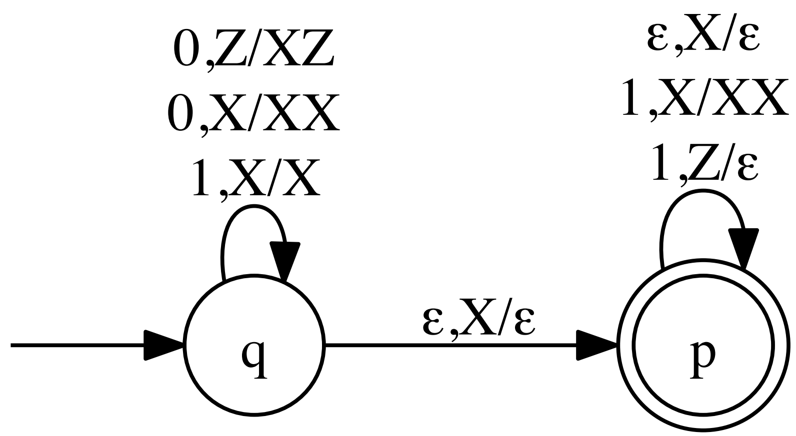
  suppose 

, then for all strings w,

$$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$$

Computation as a sequence of IDs

Example: Starting from the ID $(q, 010, Z)$, show all the reachable ID's for the PDA below.



The languages of a PDA

- For a DFA D, we saw that $L(D) = \{w : \hat{\delta}(q_0, w) \in F\}$, where F is the set of accepting states.
- For PDAs, there are two equivalent approaches to define the languages they accept:
  - Acceptance by final state
  - Acceptance by empty stack

Acceptance by final state

- Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA.
- The language accepted by P by final state is

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

Notice:

- P consumes w completely.
- P halts in an accepting state.
- The stack might not be emptied.

Acceptance by empty stack

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

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

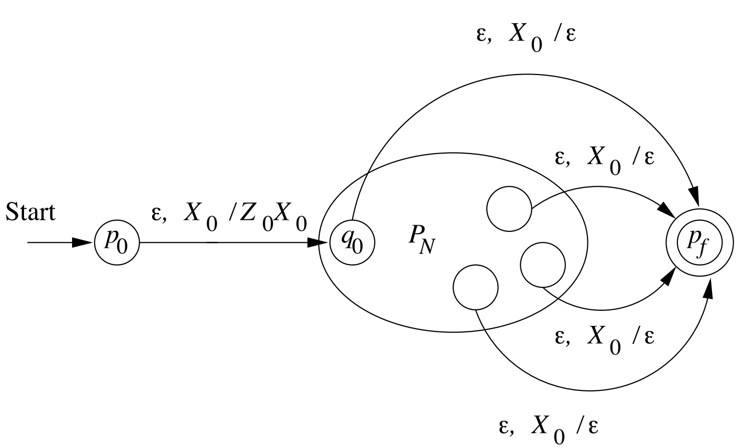
Notice:

- q can be any state
- P at the same time consumes w and empties the stack
- F is redundant in the definition of the PDA and is usually left off in the representation.

From Empty Stack to Final State

Theorem: If $P_N = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ accepts by empty stack, then $\exists$ PDA P_F that accepts by final state such that $L_e(P_N) = L_f(P_F)$.

- The idea behind the proof of this theorem is to build a PDA P_F that simulates P_N and accepts if P_N empties the stack.



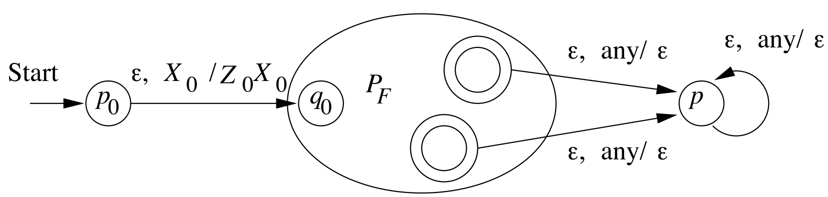
- $\delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$
- Keep all state transitions of P_N.
- Add transitions $\delta_F(q, \epsilon, X_0) = \{(p_f, \epsilon)\}$ for every state $q \in Q$

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

From Final State to Empty Stack

Theorem: If $P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ accepts by final state, then $\exists$ PDA P_N that accepts by empty stack such that $L_f(P_F) = L_e(P_N)$.

- The idea behind the proof of this theorem is to build a PDA P_N that simulates P_F and accepts if P_F accepts by final state.



- $\delta_N(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$
- Keep all state transitions of P_F.
- Add transitions $\delta_N(q, \epsilon, X) = \{(p_f, \epsilon)\}$ for every state $q \in F$ and symbol $X \in \Gamma \cup \{X_0\}$.

Equivalence of PDAs and CFGs

- We shall see that PDAs and CFGs are equivalent in power: both represent the same class of languages.
- We will do that by showing how to obtain a PDA from a CFG, and vice-versa.
- Our objective is to prove the following theorem:

  *A language is context-free **iff** some PDA recognizes it.*

  Therefore, two lemmas:

L1: If a language is context-free, then some PDA recognizes it.

L2: If a PDA recognizes some language, then it is context-free.

Proving L1

- Proof idea: show how to convert a CFG G to a PDA P.
- We will design P to simulate leftmost derivations of strings according to G.
- The stack will contain the strings of the derivations.

The derivation of the string $aaabbb$ according to the grammar below gives us a hint on the operation of the PDA.

$$A \rightarrow aAb \mid \epsilon$$

$$A \Rightarrow aAb \Rightarrow aaAbb \Rightarrow aaaAbbb \Rightarrow aaabbb$$

Proof idea of L1 (cnt.)

This is how a PDA would accept a string (by empty stack) based on the grammar rules:

- Place the start variable on the stack.
- Repeat the following steps:
  a) If the top of the stack is a variable A, nondeterministically select a rule $A \to \gamma$ and substitute A on the stack by γ.
  b) If the top of the stack is a terminal symbol a, read the next symbol from the input and compare it to a. If they don't match, reject this branch of the nondeterminism.

Obtaining a PDA from a CFG

1. For each variable A in grammar G,

$$\delta(q, \epsilon, A) = \{(q, \beta) \mid a \to \beta \text{ is a production of } G\}$$

2. For each terminal a, $\delta(q, a, a) = \{(q, \epsilon)\}$.

Example:

$$A \to aAb|\epsilon$$

Use IDs to show that the PDA accepts $aabb$ by empty stack.