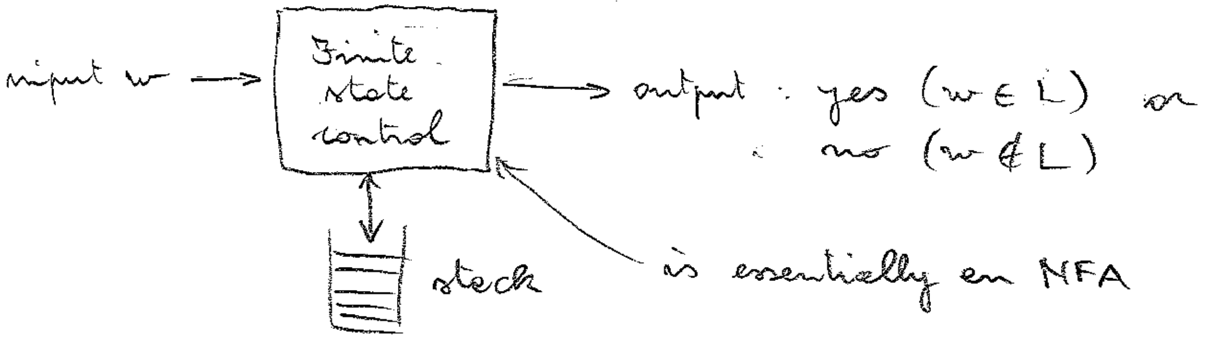
Pushdown automata

Are a class of machines corresponding to the CFLs.

- need unbounded memory to go beyond finite-state
- access to memory is restricted



- stack notation

written as $\underset{\uparrow \text{ top}}{A}BBA\underset{\uparrow \text{ bottom}}{C}$

pop : returns top symbol (e.g. A)
top symbol removed from stack (e.g. BBAC)

push(XYZ) : new content is XYZBBAC

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

- Q : states - $p, q, r, s, \ldots$
- Σ input alphabet - $a, b, c, \ldots$
- Γ stack alphabet - $A, B, C, \ldots$

} finite, nonempty sets

- q_0 : start state ($q_0 \in Q$)
- Z_0 start stack-symbol ($Z_0 \in \Gamma$)
- F final states ($F \subseteq Q$)

Note: usually $\Sigma \subseteq \Gamma$, but not necessarily

Notation: we use for strings in Σ^* : $w, x, y, z, \ldots$
Γ^* : $\alpha, \beta, \gamma, \ldots$

Transition function δ:

transitions determined by: - current state
- input (or ε-move)
- top of stack

effect: - new state
- pop, and push new string
- advance on input

$$\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \times 2^{Q \times \Gamma^*}$$

written $\delta(q, a, X) = \{(p_1, \alpha_1), \ldots, (p_k, \alpha_k)\}$

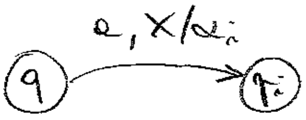
A transition is executed as follows:

1) pop stack top to determine X
2) read input to determine a (unless $a = \varepsilon$)
3) with current state q, select non-deterministically one of the pairs $(p_i, \alpha_i) \in \delta(q, a, X)$
4) change state to p_i
5) advance past a on input (unless $a = \varepsilon$)
6) push α_i on top of stack.

Note: initially, stack must contain Z_0, to allow the first transition to pop the stack. (ε is not allowed for the stack symbol)

Graphical representation as transition diagram:

if $(p_i, \alpha_i) \in \delta(q, a, X)$



Example: $L = \{0^n 1^n \mid n \geq 1\}$

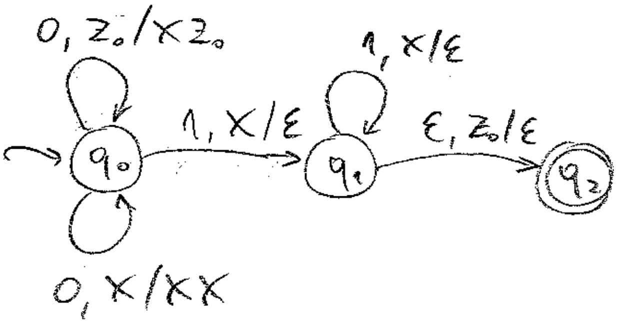
PDA M : $Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{X, Z_0\}$

$F = \{q_2\}$

transitions:



Idea:

- if input is not in $0^* 1^*$ then transition will not be defined
- if too few 1's, will not go to final state
- if too many 1's, gets stuck and cannot advance on input

Note: diagram means

$$\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$$
$$\vdots$$

<u>Instantaneous description</u> (ID):

used to describe succinctly any configuration a PDA can reach

$$ID = \langle \underset{\substack{\uparrow \\ \text{current} \\ \text{state} \\ q \in Q}}{q}, \underset{\substack{\uparrow \\ \text{unread} \\ \text{input} \\ x \in \Sigma^*}}{x}, \underset{\substack{\uparrow \\ \text{stack} \\ \text{content} \\ \alpha \in \Gamma^*}}{\alpha} \rangle$$

Transitions described using IDs:

suppose $(p_i, \alpha_i) \in \delta(q, a, X)$

we can say: $\langle q, aw, X\beta \rangle \underset{\substack{\uparrow \\ \text{'goes to' (directly)}}}{\vdash} \langle p_i, w, \alpha_i \beta \rangle$

We write $ID_1 \vdash^* ID_n$ if

$$\underset{\substack{\uparrow \\ \text{execution trace}}}{ID_1 \vdash ID_2 \vdash ID_3 \vdash \dots \vdash ID_n}$$

Acceptence:

PDA accepts w if there is at least one execution trace that leads to a final state when input is finished.

Rejects when: - no transition is possible (stuck), or
- input not over, but stack is empty, or
- input over, but not in final state

(to reject w, for every execution trace, one of these must hold)

Definition: language $\mathcal{L}(M)$ accepted by a PDA

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

$$\mathcal{L}(M) = \{ w \in \Sigma^* \mid \langle q_0, w, Z_0 \rangle \vdash^* \langle p, \varepsilon, \alpha \rangle \text{ with } p \in F, \text{ and } \alpha \in \Gamma^* \}$$

Note: nondeterminism is dealt with as for NFA's

$$\langle q_0, w, Z_0 \rangle \vdash^* \langle p, \varepsilon, \alpha \rangle$$

means that $\vdash$ can lead to $\langle p, \varepsilon, \alpha \rangle$
(provided the right nondeterministic choices are made)

$\mathcal{L}(M)$ is called final state language.

Example: PDA for $0^n 1^n$ on input 0011

initial ID: $ID_0 = \langle q_0, 0011, Z_0 \rangle$

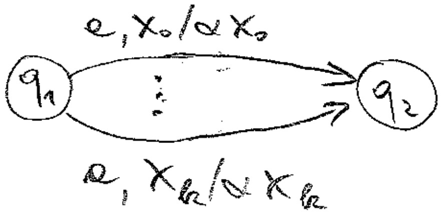
execution: $\langle q_0, 0011, Z_0 \rangle \vdash \langle q_0, 011, XZ_0 \rangle \vdash \langle q_0, 11, XXZ_0 \rangle$
$\vdash \langle q_1, 1, XZ_0 \rangle \vdash \langle q_1, \varepsilon, Z_0 \rangle \vdash \langle q_2, \varepsilon, \varepsilon \rangle$

thus $\langle q_0, 0011, Z_0 \rangle \vdash^* \langle q_2, \varepsilon, \varepsilon \rangle$

Note: • At end stack was empty, but this cannot happen inbetween, as PDA must pop at each transition. (this is why we initialise the stack with Z_0)

- We have ε-moves that ignore the input, but we must use the symbol on top of the stack
- We can simulate moves that "ignore" top of stack as follows:

suppose $\Gamma = \{X_0, \dots, X_k\}$



i.e. whatever is on top of the stack, we use it and then put it back again

We could also define a different notion of acceptance:

$$N(M) = \{ w \in \Sigma^* \mid \langle q_0, w, Z_0 \rangle \vdash^* \langle q, \varepsilon, \varepsilon \rangle \text{ for any } q \}$$

$N(M)$ is called empty stack language

Note: for $N(M)$ we ignore completely final states and accept if at the end of the input word the stack is empty

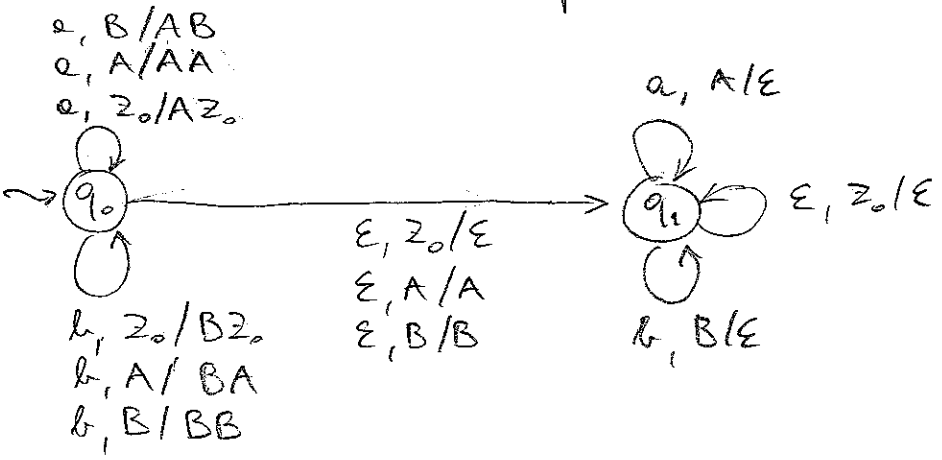
Reject when, for every execution trace, one of the following happens:
- before w is over, PDA gets stuck
- – " – stack is empty
- when – " – stack is not empty

Example: $L_{epal} = \{ww^R \mid w \in \{a, b\}^*\}$

M: $Q = \{q_0, q_1\}$ $\quad \Gamma = \{A, B, Z_0\}$

$\Sigma = \{a, b\}$ $\quad F = \emptyset$

We want that $\mathcal{N}(M) = L_{epal}$



Idea:
1) push w onto stack one by one, staying in q_0
2) guess mid-point and move to q_1
3) in q_1, match remaining input with stack one by one
4) at end remove Z_0 from top of stack to accept

Note: in 3, stack pops w in reverse order

Exercise E6.1: Construct execution trace that shows acceptance of abbbba

Exercise E6.2 Construct M' s.t. $\mathcal{L}(M') = L_{epal}$
acceptance by final state

Exercise E6.3 Let $L_{pal} = \{w \in \{a, b\}^* \mid w \text{ is a palindrome}\}$

Construct M^{f}_{pal} s.t. $\mathcal{L}(M^{f}_{pal}) = L_{pal}$

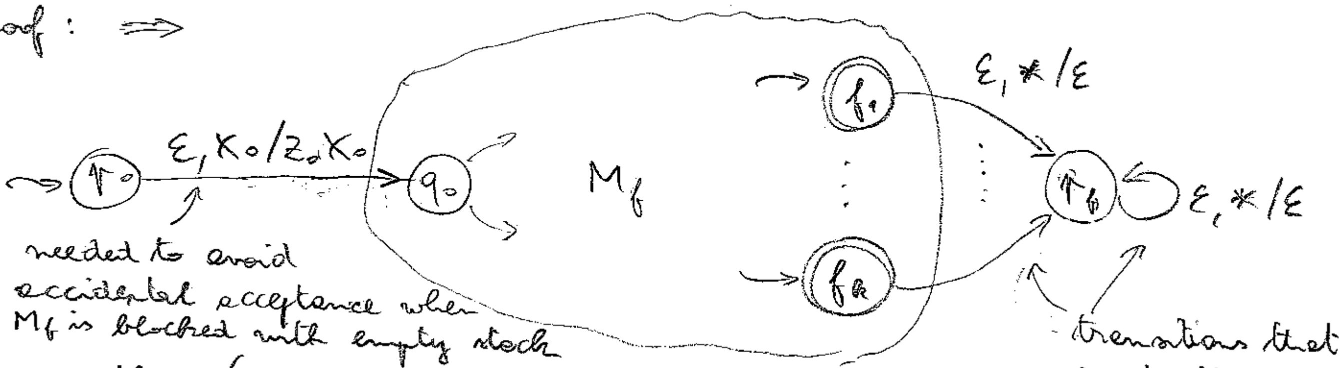
–"– M^{es}_{pal} s.t. $\mathcal{N}(M^{es}_{pal}) = L_{pal}$

The two acceptance conditions give rise to automata with the same expressive power

Theorem: $L = \mathcal{L}(M_f)$ for some PDA M_f $\iff$

$L = \mathcal{N}(M_{es})$ for some PDA M_{es}

Proof: $\Rightarrow$



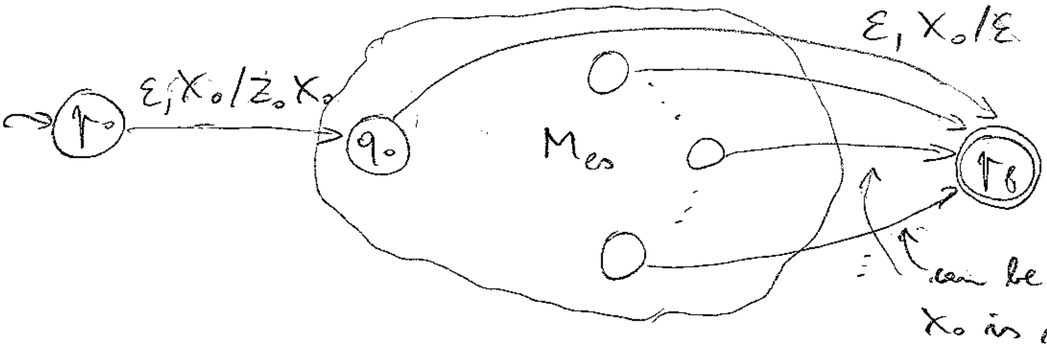
needed to avoid accidental acceptance when M_f is blocked with empty stack

transitions that empty the stack

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

$M_{es} = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_{es}, p_0, X_0, \emptyset)$

$\Leftarrow$



can be done only when X_0 is seen on the stack i.e., when M_{es} would have emptied it

$M_{es} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$

$M_f = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_f, p_0, X_0, \{p_f\})$

Question: Why did we introduce two acceptance conditions

$\mathcal{L}(M)$ ... resembles acceptance condition of FSAs

$\mathcal{N}(M)$ ... useful to show equivalence between PDAs and CFGs

Relationship between PDAs and CFGs:

Let: $\mathcal{L}(CFG)$ be the class of languages defined by CFGs

$\mathcal{L}(PDA)$ be the class of final-state languages of PDAs

$\mathcal{N}(PDA)$ — " — empty-stack — " —

By the previous theorem we know that

$$\mathcal{L}(PDA) = \mathcal{N}(PDA)$$

Theorem: $\mathcal{L}(CFG) = \mathcal{N}(PDA)$

Hence: $CFL = \mathcal{L}(CFG) = \mathcal{N}(PDA) = \mathcal{L}(PDA)$

We only sketch the proof of: $\mathcal{L}(CFG) \subseteq \mathcal{N}(PDA)$
(for the details and the other direction, see textbook)

Let G be a CFG. We want to construct a PDA M such that $\mathcal{L}(CFG) = \mathcal{N}(PDA)$

Idea: M simulates l.m. derivations of G for input w such that at any derivation step the sentential form is represented by

- sequence of symbols of w already consumed by M
- followed by contents of M's stack

e.g. In G: $S \overset{*}{\underset{l.m}{\Longrightarrow}} abXYcZ \overset{*}{\underset{l.m}{\Longrightarrow}} abxycz$

In M: $\langle q_0, abxycz, Z_0 \rangle$

consumes prefix ab $\rightarrow \vdash^* \langle q_0, xycz, XYcZ \rangle$

$\vdash^* \langle q_0, \varepsilon, \varepsilon \rangle$

Construction of M:

Let $G = (V_N, V_T, P, S)$

We define $M_G = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$

with $Q = \{q_0\}$, i.e. PDA has a single state

$\Sigma = V_T$

$\Gamma = V_T \cup V_N$

$Z_0 = S$

In δ we have two types of transitions:

1) When $a \in V_T$ is on top of stack, we expect to see a in input and consume both (sentential form of grammar derivation is not changed)

$$\Rightarrow \forall a \in V_T : \quad \delta(q_0, a, a) = \{(q_0, \varepsilon)\}$$

2) When $A \in V_N$ is on top of stack, then replace it with r.h.s of some production for A in P (no input is consumed)

$$\Rightarrow \forall A \in V_N : \text{ if } A \to \alpha_1 | \cdots | \alpha_k \text{ are in } P$$

$$\text{then } \delta(q_0, \varepsilon, A) = \{(q_0, \alpha_1), \ldots, (q_0, \alpha_k)\}$$

We can show: $\forall y \in V_T^*$, $\forall \beta \in V^*$

$S \overset{*}{\Rightarrow} y\beta$ in G iff

$\langle q_0, yx, S\rangle \vdash^* \langle q_0, x, \beta\rangle$ for all $x \in V_T^*$

(By induction on derivation length)

For $\beta = \varepsilon$ and $x = \varepsilon$ we get the claim q.e.d.