From lecture 8:

Definition

regular grammar (or *right-linear grammar*)
productions are of the form
– $A \rightarrow \sigma B$ variables A, B, terminal σ
– $A \rightarrow \Lambda$ variable A

Theorem

A language L is regular,
if and only if there is a regular grammar generating L.

Proof. . .

[M] Def 4.13, Thm 4.14

From lecture 9:

Definition

CFG in *Chomsky normal form*

productions are of the form

– $A \rightarrow BC$ variables A, B, C

– $A \rightarrow \sigma$ variable A, terminal σ

[M] Def 4.29

Outlook

Chomsky NF for pumping lemma (later)

Operations on languages

$\text{even}(L) = \{ w \in L \mid |w| \text{ even} \}$

idea: new variables for even/odd length strings
Chomsky normalform to reduce number of possibilities.

grammar $G = (V, \Sigma, P, S)$ for L, in ChNF
new grammar $G = (V', \Sigma, P', S')$ for $\text{even}(L)$
variables: $V' = \{X_e, X_o \mid X \in V\}$
axiom: $S' = S_e$
productions: – for every $A \to BC$ in P we have in P':

$$A_e \to B_e C_e \mid B_o C_o \qquad A_o \to B_e C_o \mid B_o C_e$$

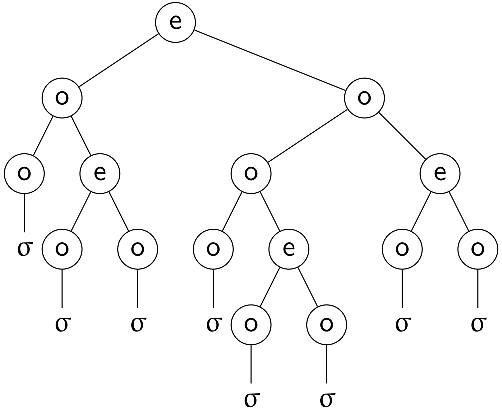
– for every $A \to \sigma$ in P we have in P': $A_o \to \sigma$

ABOVE

We consider closure properties: given an operation X show that whenever L is regular/context-free, then also $X(L)$ is regular/context-free.

This is done as follows: if L is regular/context-free, then we know there is a regular/context-free grammar G for L, and we show how to construct a new grammar G' (of the same type) for $X(L)$, in terms of the original grammar G.

Even/odd markings



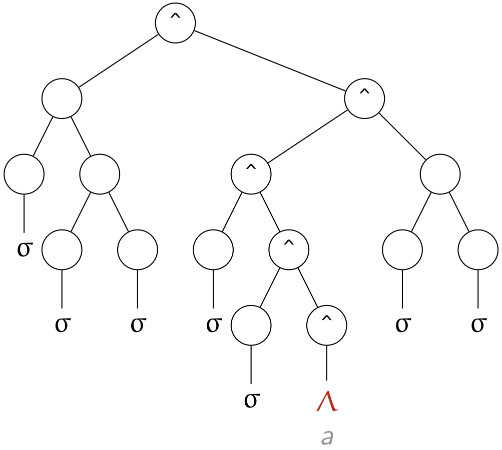
Operations on languages (2)

$L \subseteq \{a,b\}^*$, $\quad \mathrm{chop}(L) = \{\, xy \mid xay \in L \}$ $\quad$ remove some a in each string

idea: new variables for the task of removing letter a

grammar $G = (V, \{a,b\}, P, S)$ for L, in ChNF
new grammar $G = (V', \{a,b\}, P', S')$ for $\mathrm{chop}(L)$
variables: $V' = V \cup \{\hat{X} \mid X \in V\}$
axiom: $S' = \hat{S}$
productions: keep all productions from P, and
– for every $A \to BC$ add $\hat{A} \to \hat{B}C \mid B\hat{C}$
– for every $A \to a$ add $\hat{A} \to \Lambda$

Chop markings



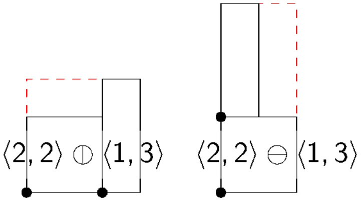
⊠Attribute grammars

$$
\begin{aligned}
E &\to E + T \mid T \\
T &\to T * F \mid F \\
F &\to (\, E\,) \mid \mathit{int}
\end{aligned}
$$

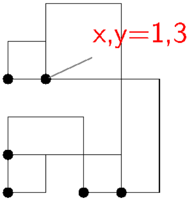
$E \to E_1 + T_1$	$E.\mathrm{val} = \mathrm{E}_1.\mathrm{val} + \mathrm{T}_1.\mathrm{val}$
$E \to T_1$	$E.\mathrm{val} = \mathrm{T}_1.\mathrm{val}$
$T \to T_1 * F_1$	$T.\mathrm{val} = \mathrm{T}_1.\mathrm{val} \cdot \mathrm{F}_1.\mathrm{val}$
$T \to F_1$	$T.\mathrm{val} = \mathrm{F}_1.\mathrm{val}$
$F \to (\, E_1\,)$	$F.\mathrm{val} = \mathrm{E}_1.\mathrm{val}$
$F \to \mathit{int}$	$F.\mathrm{val} = \mathrm{IntVal}(\,\mathrm{int}\,)$

D.E. Knuth. Semantics of Context-Free Languages.
Math. Systems Theory (1968) 127–145 doi:10.1007/BF01692511

⊠Box grammar



$\langle 2,2\rangle \obar \langle 1,3\rangle$
$\langle 2,2\rangle \ominus \langle 1,3\rangle$

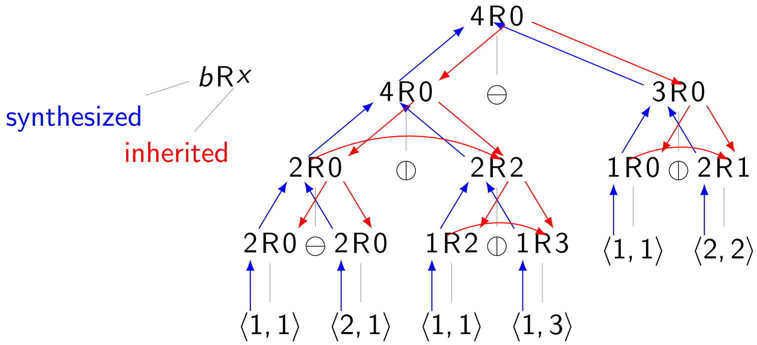


$$(\, (\langle 1,1\rangle \ominus \langle 2,1\rangle) \obar (\langle 1,1\rangle \obar \langle 1,3\rangle) \,) \ominus (\langle 1,1\rangle \obar \langle 2,2\rangle)$$

production	semantic rule
$R \to \langle E_1, E_2\rangle$	$R.b = E_1.\mathrm{val} \quad \mathrm{R.h} = \mathrm{E_2.val}$
$R \to (R_1 \obar R_2)$	$R.b = R_1.b + R_2.b$
	$R.h = \max\{R_1.h, R_2.h\}$
	$R_1.x = R.x \quad R_2.x = R.x + R_1.b$
	$R_1.y = R.y \quad R_2.y = R.y$
$R \to (R_1 \ominus R_2)$	$R.b = \max\{R_1.b, R_2.b\}$
	$R.h = R_1.h + R_2.h$
	$R_1.x = R.x \quad R_2.x = R.x$
	$R_1.y = R.y \quad R_2.y = R.y + R_1.h$

☒Evaluating attributes

$$R \to (R_1 \oplus R_2) \quad R.b = R_1.b + R_2.b$$
$$R_1.x = R.x \quad R_2.x = R.x + R_1.b$$
$$R \to (R_1 \ominus R_2) \quad R.b = \max\{R_1.b, R_2.b\}$$
$$R_1.x = R.x \quad R_2.x = R.x$$



Section 5

Pushdown Automata

Chapter

just like FA, PDA accepts strings / language
just like FA, PDA has states
just like FA, PDA reads input one letter at a time
unlike FA, PDA has auxiliary memory: a stack
unlike FA, by default PDA is nondeterministic
unlike FA, by default Λ-transitions are allowed in PDA

Why a stack?

$$AnBn = \{a^i b^i \mid i \geqslant 0\}$$

with $x = aaabbb$

$$SimplePal = \{xcx^r \mid x \in \{a, b\}^*\}$$

with $x = aabcbaa$

Stack in PDA contains symbols from certain alphabet.
Usual stack operations: pop, top, push
Extra possiblity: replace top element X by string α

AnBn

$AnBn = \{ a^n b^n \mid n \geqslant 0 \}$

PDA...

[M] E 5.3

AnBn

$AnBn = \{\ a^n b^n \mid n \geqslant 0\ \}$

initial q_0, Z_0, accept $A = \{q_0, q_3\}$

$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[r, "{a, Z_0/aZ_0}"] & q_1 \arrow[loop above, "{a, a/aa}"] \arrow[r, "{b, a/\Lambda}"] & q_2 \arrow[loop above, "{b, a/\Lambda}"] \arrow[r, "{\Lambda, Z_0/Z_0}"] & q_3
\end{tikzcd}$$

[M] E 5.3

Stack in PDA contains symbols from certain alphabet.
Usual stack operations: pop, top, push
Extra possiblity: replace top element X by string α

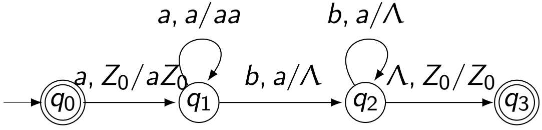
$\alpha = \Lambda$	pop
$\alpha = X$	top
$\alpha = YX$	push
$\alpha = \beta X$	push*
$\alpha = \dots$	

Top element X is required to do a move!

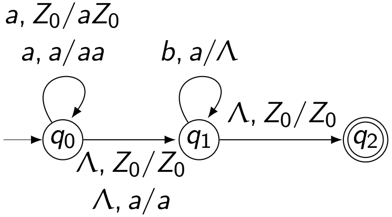
AnBn

$AnBn = \{ a^n b^n \mid n \geqslant 0 \}$

initial q_0, Z_0, accept $A = \{q_0, q_3\}$

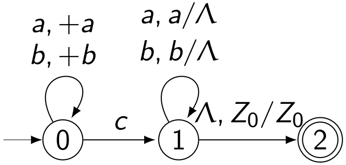


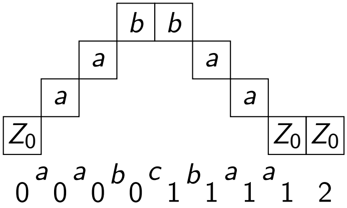
[M] E 5.3



Using a stack/pushdown

$SimplePal =$
$\{ \, xcx^r \mid x \in \{a, b\}^* \, \}$





Z_0 a a b b a a Z_0 Z_0
0 a 0 a 0 b 0 c 1 b 1 a 1 a 1 2

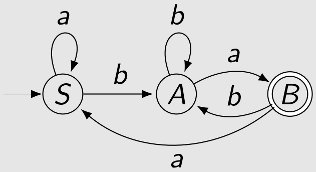
[M] Fig 5.5

Regular languages and CF grammars

From lecture 8:

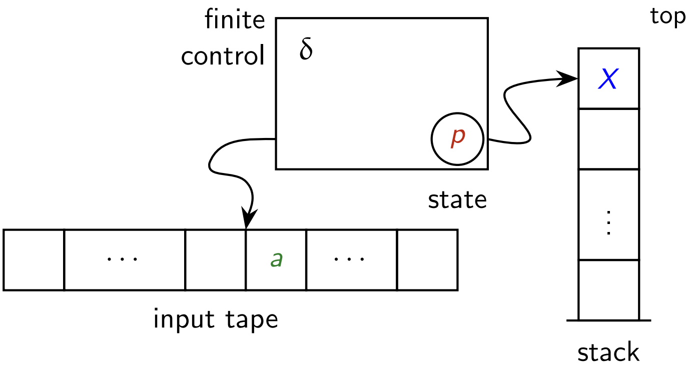
systematic approach

Example



axiom S	initial state
$S \to bA \mid aS$	transitions
$A \to bA \mid aB$	
$B \to bA \mid aS$	
$B \to \Lambda$	accepting state

Intuition



Pushdown automaton

Definition

PDA 7-tuple $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$

Q	*states*	p, q	
Σ	*input alphabet*	a, b	w, x
Γ	*stack alphabet*	a, b, A, B	α
$q_0 \in Q$	*initial state*		
$Z_0 \in \Gamma$	*initial stack symbol*		
$A \subseteq Q$	*accepting states*		

$\delta : \ldots \to \ldots$

transition function

$$\underbrace{(\overset{\text{from}}{p} \quad \underset{\text{read}}{a} \quad \underset{\text{pop}}{X}}_{\text{before}} \quad \underbrace{\overset{\text{to}}{q} \quad \underset{\text{push}}{\alpha}}_{\text{after}} \quad)$$

Pushdown automaton

Definition

PDA 7-tuple $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$

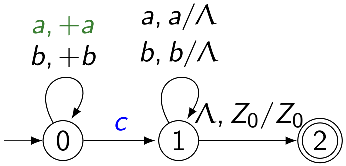
Q	*states*	p, q	
Σ	*input alphabet*	a, b	w, x
Γ	*stack alphabet*	a, b, A, B	α
$q_0 \in Q$	*initial state*		
$Z_0 \in \Gamma$	*initial stack symbol*		
$A \subseteq Q$	*accepting states*		

$$\delta : Q \times (\Sigma \cup \{\Lambda\}) \times \Gamma \to 2^{Q \times \Gamma^*}$$

transition function (finite)

In principle, Z_0 may be removed from the stack, but often it isn't.

$SimplePal = \{ xcx^r \mid x \in \{a, b\}^* \}$



$Q = \{0, 1, 2\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{a, b, Z_0\}$
$q_0 = 0$
$Z_0 = Z_0$
$A = \{2\}$

Transition table:

State p	**Input** σ	**Stack Symbol** X	**Move(s)** $\delta(p, \sigma, X)$
0	a	Z_0	$(0, aZ_0)$
0	a	a	$(0, aa)$
0	a	b	$(0, ab)$
0	b	Z_0	$(0, bZ_0)$
0	b	a	$(0, ba)$
0	b	b	$(0, bb)$
0	c	Z_0	$(1, Z_0)$
0	c	a	$(1, a)$
0	c	b	$(1, b)$
1	a	a	$(1, \Lambda)$
1	b	b	$(1, \Lambda)$
1	Λ	Z_0	$(2, Z_0)$
(all other combinations)			none

Pushing and popping

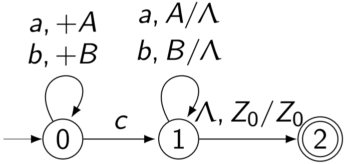
transition $(q, \alpha) \in \delta(p, a, A)$ $$\begin{tikzcd} p \arrow[r, "{a, A/\alpha}"] & q \end{tikzcd}$$

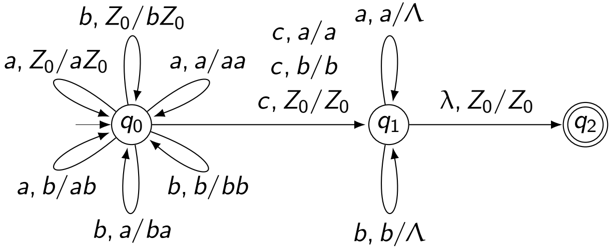
$(p, a, A) \mapsto (q, \alpha)$

$p, q \in Q,\ a \in \Sigma \cup \{\Lambda\},\ A \in \Gamma,\ \alpha \in \Gamma^*$

intuitive	formalized as		convention
pop A	$(q, \Lambda) \in \delta(p, a, A)$	$\alpha = \Lambda$	$p \xrightarrow{a, A/\Lambda} q$
push A	$(q, AX) \in \delta(p, a, X)$	for all $X \in \Gamma$	$p \xrightarrow{a, +A} q$
read a	$(q, X) \in \delta(p, a, X)$	for all $X \in \Gamma$	$p \xrightarrow{a} q$

Differences in dialect





[M] Fig 5.5

ABOVE

The 'same' PDA twice. First in the version of HJH where we allow some shortcuts in notation.

Second as depicted in the book. Note Martin happily pushes terminals like a, b on the stack. Formally that is OK

Computation and language

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

configuration (q, x, α) $\quad q \in Q,\ x \in \Sigma^*,\ \alpha \in \Gamma^*$

state, remaining input, stack with top left

step $(p, ax, B\alpha) \vdash_M (q, x, \beta\alpha)$ $\quad$ when $(q, \beta) \in \delta(p, a, B)$

$\vdash^n_M \qquad \vdash^*_M \qquad \vdash \qquad \vdash^n \qquad \vdash^*$

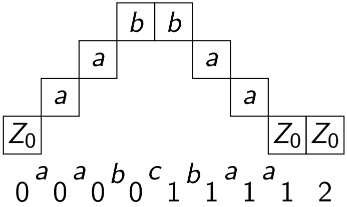
Definition

String x accepted by M (by *final state*), if
$(q_0, x, Z_0) \vdash^* (q, \Lambda, \alpha)$ for some $q \in A$, and some $\alpha \in \Gamma^*$
Language accepted by M (by *final state*)
$L(M) = \{\, x \in \Sigma^* \mid x \text{ accepted by } M \,\}$

read complete input, end in accepting state, some path

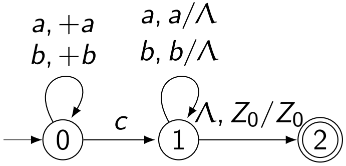
[M] D 5.2

Using a stack/pushdown



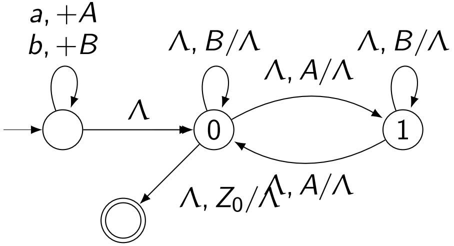
b b
a a
a a
Z_0 Z_0 Z_0
$0 \overset{a}{} 0 \overset{a}{} 0 \overset{b}{} 0 \overset{c}{} 1 \overset{b}{} 1 \overset{a}{} 1 \overset{a}{} 1 \quad 2$

$\mathit{SimplePal} =$
$\{\, xcx^R \mid x \in \{a, b\}^* \,\}$



$$
\begin{array}{l}
(0, aabcbaa, Z_0) \vdash \\
(0, abcbaa, aZ_0) \vdash \\
(0, bcbaa, aaZ_0) \vdash \\
(0, cbaa, baaZ_0) \vdash \\
(1, baa, baaZ_0) \vdash \\
(1, aa, aaZ_0) \vdash \\
(1, a, aZ_0) \vdash \\
(1, \Lambda, Z_0) \vdash \\
(2, \Lambda, Z_0)
\end{array}
$$

[M] Fig 5.5



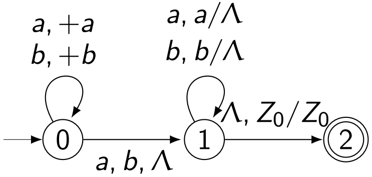
$a, +A$
$b, +B$
Λ
$\Lambda, B/\Lambda$
$\Lambda, A/\Lambda$
$\Lambda, B/\Lambda$
0
1
$\Lambda, A/\Lambda$
$\Lambda, Z_0/\Lambda$

ABOVE

Λ-computations can be very long in PDA, they can even loop.
In the example the input is read and stored on the tape, and at the end of the input it is verified that the string contains an even number of a's.

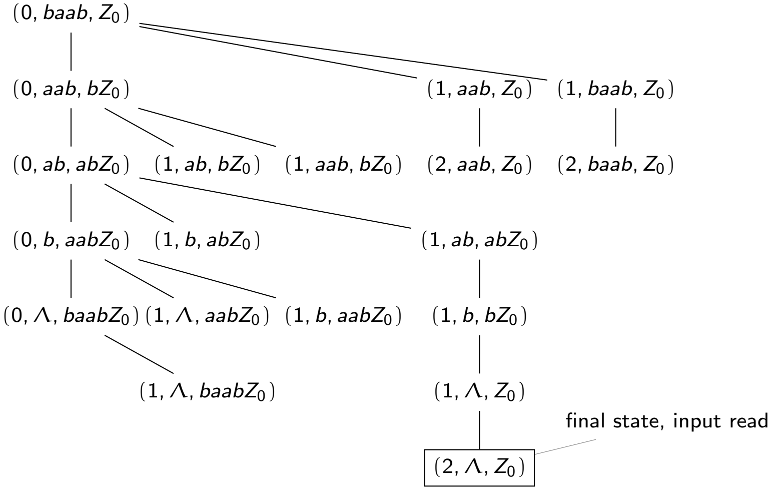
Pal $\{ y \in \{a, b\}^* \mid y = y^r \}$

Pal $\{ y \in \{a, b\}^* \mid y = y^r \}$



$Q = \{0, 1, 2\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{a, b, Z_0\}$
$q_0 = 0$
$Z_0 = Z_0$
$A = \{2\}$

Computation tree



[M] Fig 5.9

ABOVE

Non-determinism at work. The PDA for palindromes cannot see what is the middle of the input string, and has to guess. Only one of the guesses leads to an accepting computation.

Deterministic PDA

for each state and stack symbol
– on each symbol/Λ at most one transition
– not both symbol and Λ-transition

Definition

$\delta(q, \sigma, X) \cup \delta(q, \Lambda, X)$ at most one element for each $q \in Q, \sigma \in \Sigma, X \in \Gamma$

[M] Def 5.10

DPDA for *Balanced*

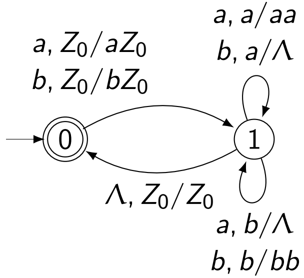
$\mathit{Balanced} = \{$balanced strings of brackets [and]$\}$

[M] E 5.11

DPDA for AeqB

[M] E 5.13

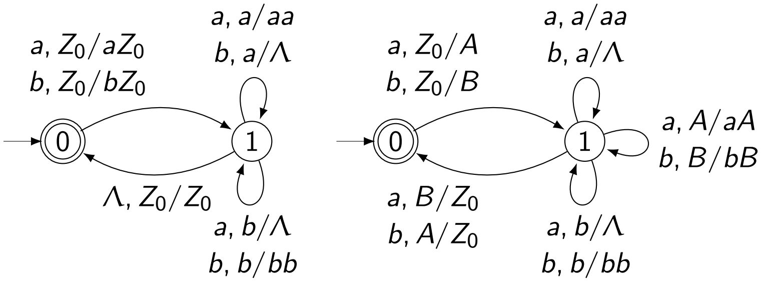
DPDA for AeqB



Without Λ-transitions...

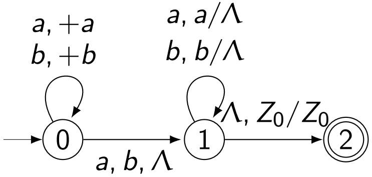
[M] E 5.13

DPDA for AeqB



[M] E 5.13

Pal $\{ y \in \{a, b\}^* \mid y = y^r \}$



$Q = \{0, 1, 2\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{a, b, Z_0\}$
$q_0 = 0$
$Z_0 = Z_0$
$A = \{2\}$

Theorem

The language Pal *cannot be accepted by a deterministic pushdown automaton.*

Proof. . .

[M] Thm 5.16

Distinguishing strings

From lecture 3:

Definition

Let L be language over Σ, and let $x, y \in \Sigma^*$.
Then x, y are *distinguishable* wrt L (*L-distinguishable*),
if there exists $z \in \Sigma^*$ with

$$xz \in L \text{ and } yz \notin L \quad \text{or} \quad xz \notin L \text{ and } yz \in L$$

Such z *distinguishes* x and y wrt L.

[M] D 2.20

Pal

From lecture 3:
$Pal = \{x \in \{a, b\}^* \mid x = x^r\}$

For Every Pair x, y of Distinct Strings in $\{a, b\}^*$, x and y Are Distinghuishable with Respect to Pal.

[M] E. 2.27

Theorem

The language Pal *cannot be accepted by a deterministic pushdown automaton.*

Proof.

Assume M is DPDA for Pal.

No assumption on form transitions M.

M reads every string $x \in \{a, b\}^*$ completely, with one path.

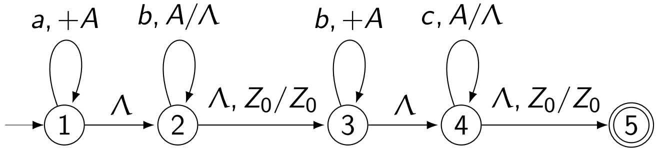
There exist different strings $r, s \in \{a, b\}^*$, such that for every $z \in \{a, b\}^*$, M treats rz and sz the same way.

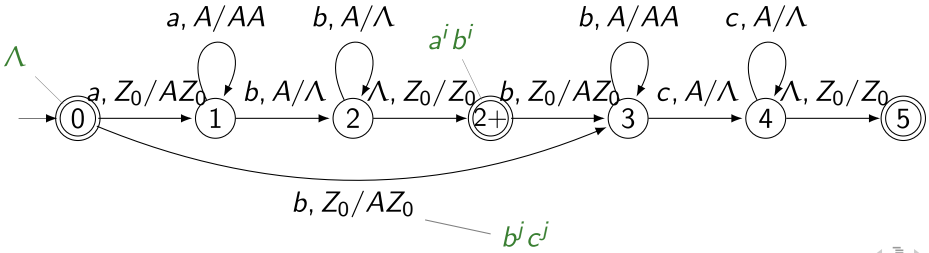
There exist different strings $r, s \in \{a, b\}^*$, such that for every $z \in \{a, b\}^*$, M treats rz and sz the same way.

For a string $x \in \{a, b\}^*$, let y_x be a string such that height of stack after xy_x is minimal.
Let α_x be stack after xy_x.
(state, top stack symbol) determines how suffix z is treated.
Infinitely many strings xy_x. Why?
Finitely many pairs (q, X)
Different $r = uy_u$ and $s = vy_v$ arrive at same pair (q, X).
For any suffix z, rz and sz are treated the same:
$rz \in Pal \iff sz \in Pal$.
Contradiction.

$a^i b^j c^k \quad j = i + k$

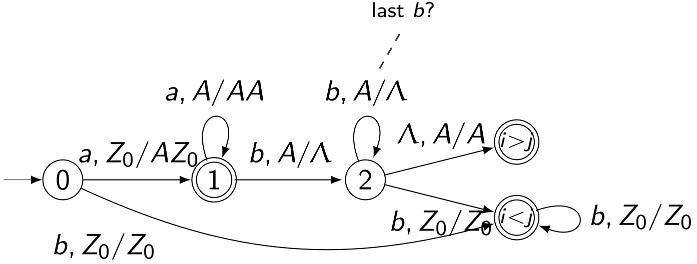
$S \to AB$
$A \to aAb \mid \Lambda$
$B \to bBc \mid \Lambda$

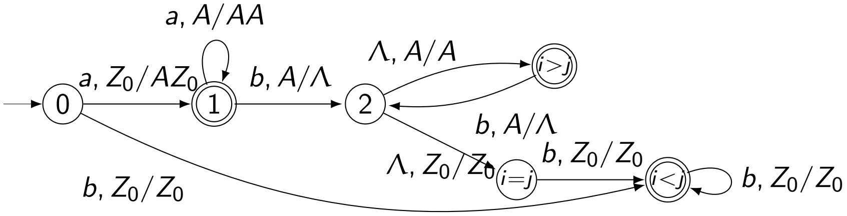




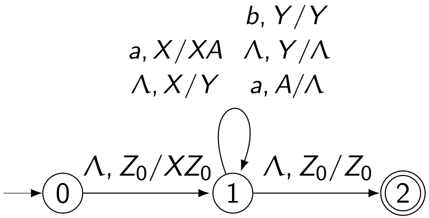
AnB-not-n

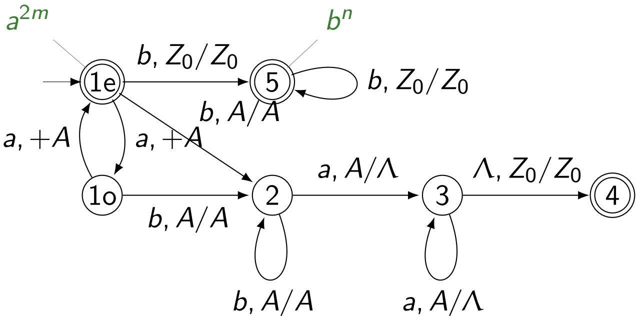
$\{ \, a^i b^j \mid i \neq j \, \}$





$a^m b^n a^m \qquad m, n \geqslant 0$





ABOVE

The first PDA is not deterministic. Actually it is working like a grammar: in state 1 the following productions are simulated:

$X \to aXA \mid Y$

$Y \to bY \mid \Lambda$

$A \to a$

The second automaton is deterministic. We have to distinguish the cases where $m = 0$ (state 5) and $n = 0$ (states $1e$ and $1o$).