Huiswerk

- Deadline huiswerkopgave 1: vrijdag 9 oktober, 23.59

From lecture 2:

Definition

regular grammar (or *right-linear grammar*)
productions are of the form
– $A \to \sigma B$ variables A, B, terminal σ
– $A \to \Lambda$ 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 4:

Definition

CFG in *Chomsky normal form*
productions are of the form
– $A \to BC$ variables A, B, C
– $A \to \sigma$ variable A, terminal σ

Theorem

For every CFG G there is CFG G_1 in CNF such that $L(G_1) = L(G) - \{\Lambda\}$.

[M] Def 4.29, Thm 4.30
What about Λ...

Chomsky NF for pumping lemma (later)

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

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

grammar $G = (V, \Sigma, P, S)$ for L, in ChNF
new grammar $G' = (V', \Sigma, P', S')$ for $\text{even}(L)$

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

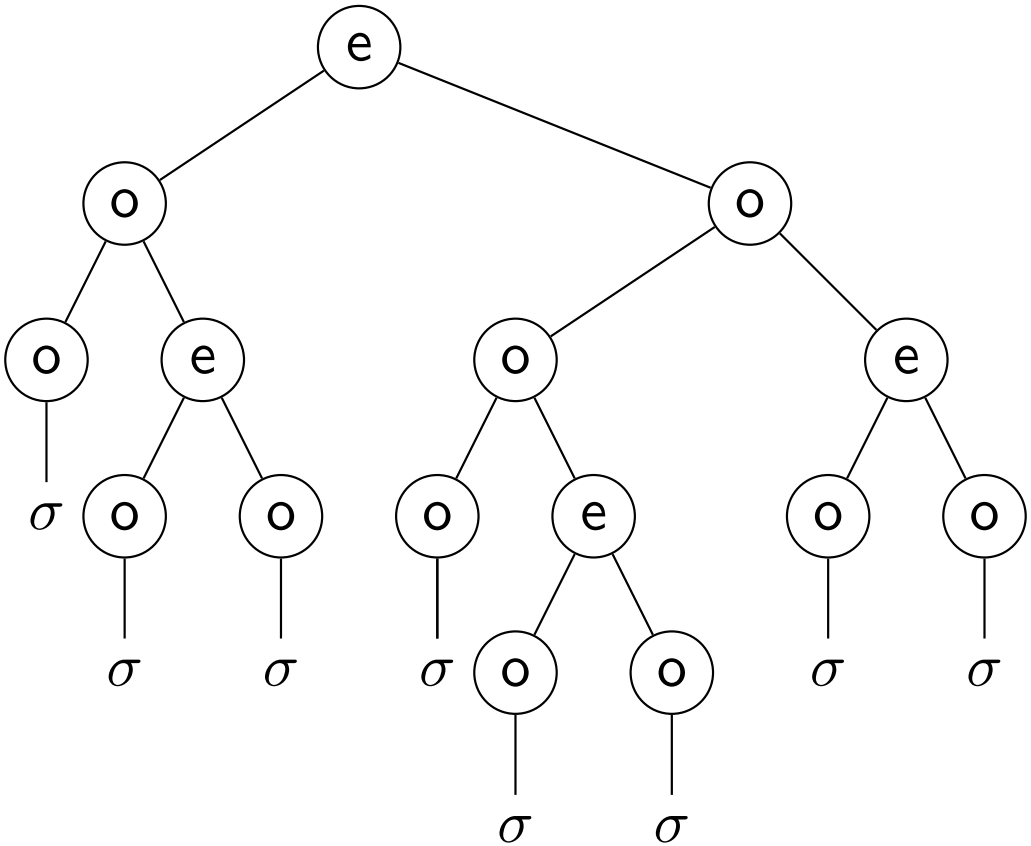
idea: new variables for even/odd length strings
Chomsky normal form to reduce number of possibilities.

grammar $G = (V, \Sigma, P, S)$ for L, in ChNF
new grammar $G' = (V', \Sigma, P', S')$ for 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': $A_o \to \sigma$

Even/odd markings



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.

Operations on languages (2)

$L \subseteq \{a, b\}^*$, $\quad \text{chop}(L) = \{\, xy \mid xay \in L \}$ $\quad$ 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 $\text{chop}(L)$

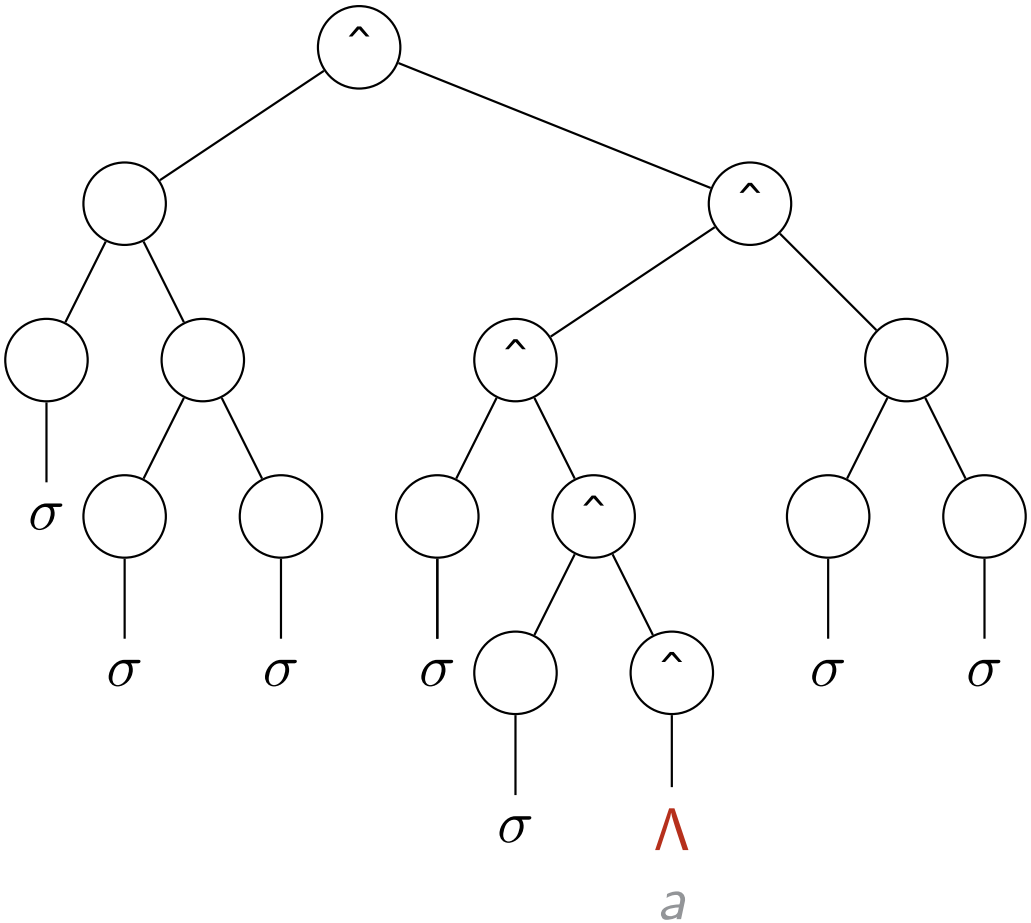
Operations on languages (2)

$L \subseteq \{a, b\}^*$, $\quad \text{chop}(L) = \{\, xy \mid xay \in L \}$ $\quad$ 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 $\text{chop}(L)$
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

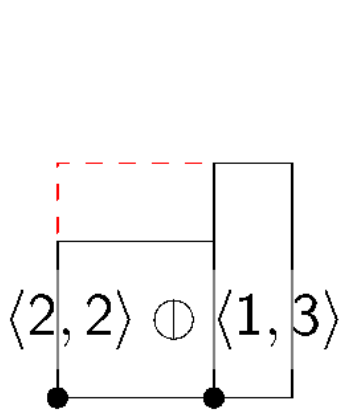


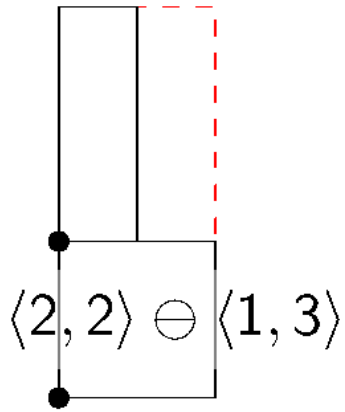
$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (E) \mid int$$

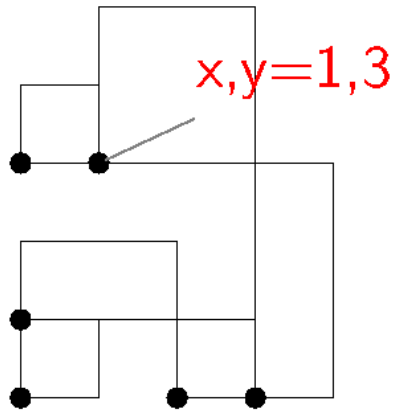
$$\begin{array}{ll} E \to E_1 + T_1 & E.\mathrm{val} = \mathrm{E}_1.\mathrm{val} + \mathrm{T}_1.\mathrm{val} \\ E \to T_1 & E.\mathrm{val} = \mathrm{T}_1.\mathrm{val} \\ T \to T_1 * F_1 & T.\mathrm{val} = \mathrm{T}_1.\mathrm{val} \cdot \mathrm{F}_1.\mathrm{val} \\ T \to F_1 & T.\mathrm{val} = \mathrm{F}_1.\mathrm{val}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

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

☒Box grammar







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

production	semantic rule
$R \to \langle E_1, E_2\rangle$	$R.b = E_1.\text{val} \quad \text{R.h} = \text{E}_2.\text{val}$
$R \to (R_1 \oplus R_2)$	$R.b = R_1.b + R_2.b$
	$R.h = \max\{R_1.h, R_2.h\}$
	$R_1.x = R.x \quad R_2.x = R.x + R_1.b$
	$R_1.y = R.y \quad R_2.y = R.y$
$R \to (R_1 \ominus R_2)$	$R.b = \max\{R_1.b, R_2.b\}$
	$R.h = R_1.h + R_2.h$
	$R_1.x = R.x \quad R_2.x = R.x$
	$R_1.y = R.y \quad R_2.y = R.y + R_1.h$

$R \to (R_1 \oplus R_2)$ $\quad R.b = R_1.b + R_2.b$

$\quad\quad\quad

Section 5

Pushdown Automata

2 Pushdown Automata
- Deterministic PDA
- From CFG to PDA
- From PDA to CFG
- Empty stack acceptance
- LL(1)

reg. languages	FA	reg. grammar	reg. expression
determ. cf. languages	DPDA		
cf. languages	PDA	cf. grammar	
cs. languages	LBA	cs. grammar	
re. languages	TM	unrestr. grammar	

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?

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

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

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

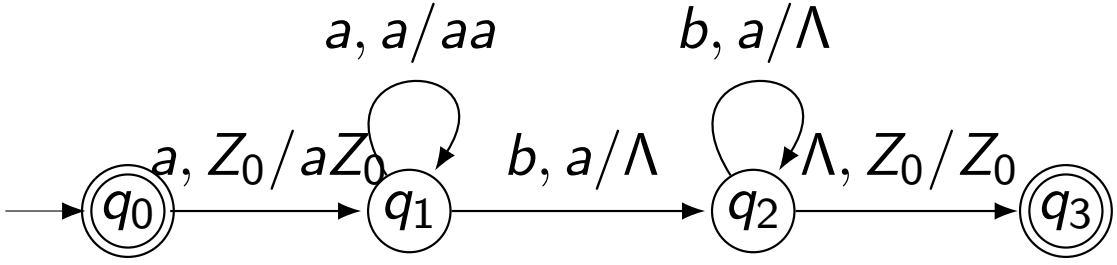
initial q_0, Z_0

PDA...

[M] E 5.3

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

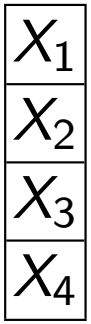
initial q_0, Z_0, accept $A = \{q_0, q_3\}$



[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

Notation:
If stack contents is $X_1X_2X_3X_4$, then top element is X_1.
If we replace X by string α, then first symbol of α ends up at top of stack.

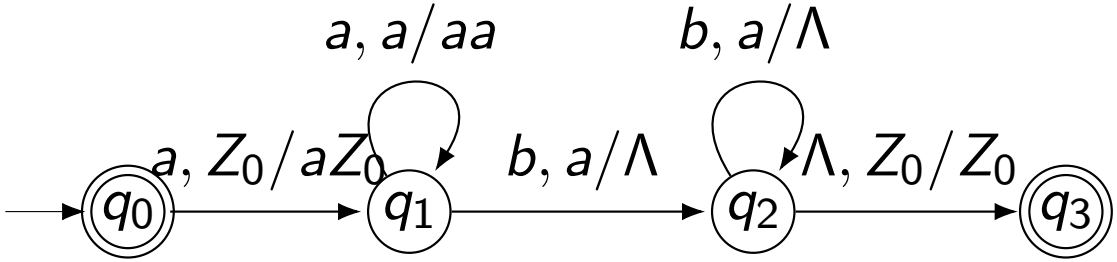


$\alpha = \Lambda$	pop
$\alpha = X$	top
$\alpha = YX$	push
$\alpha = \beta X$	push*
$\alpha = \ldots$	

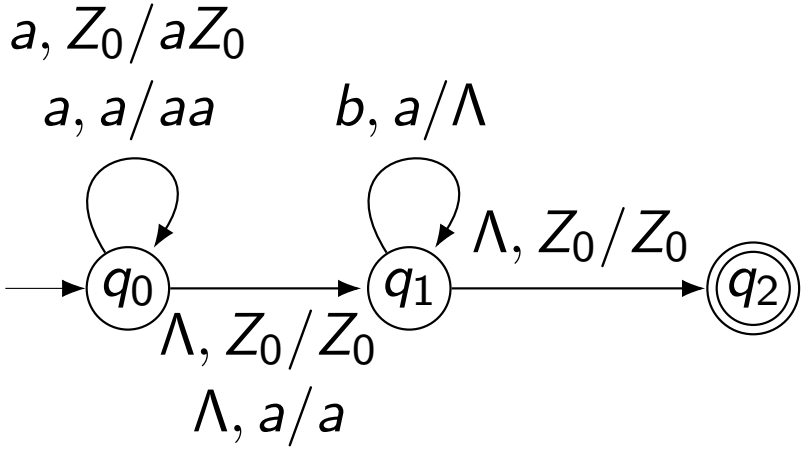
Top element X is required to do a move!

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

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



[M] E 5.3

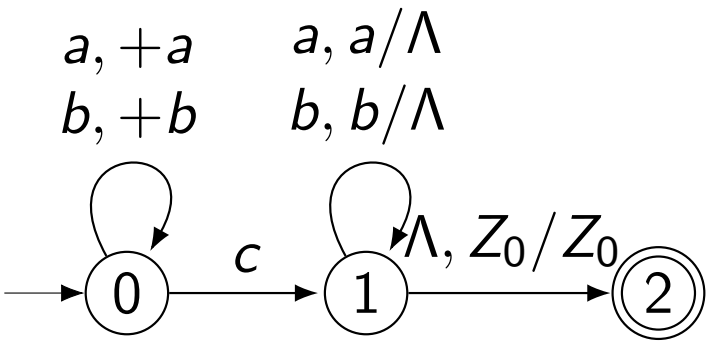


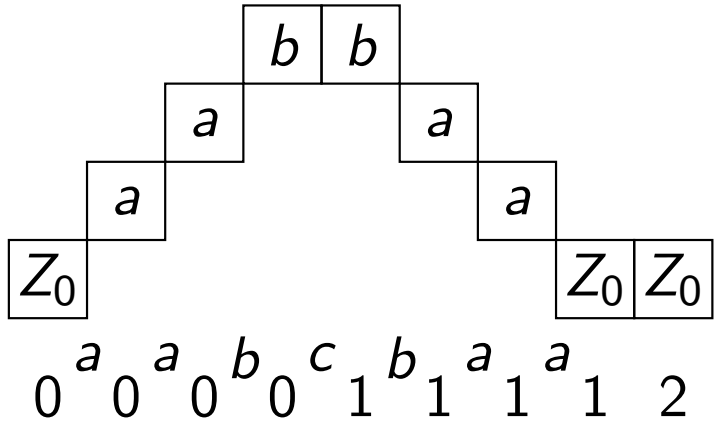
Using a stack/pushdown

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

[M] Fig 5.5

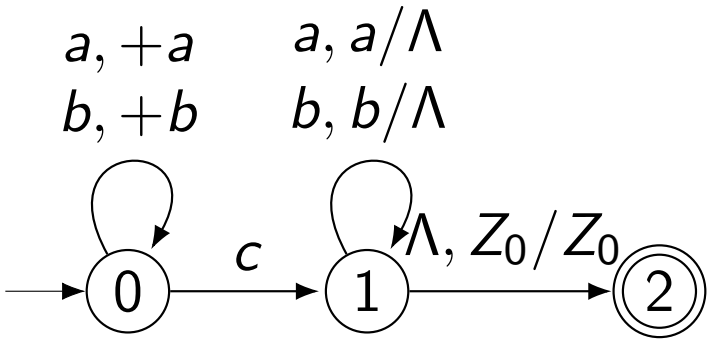
$SimplePal =$
$\{\ xcx^r \mid x \in \{a, b\}^*\ \}$



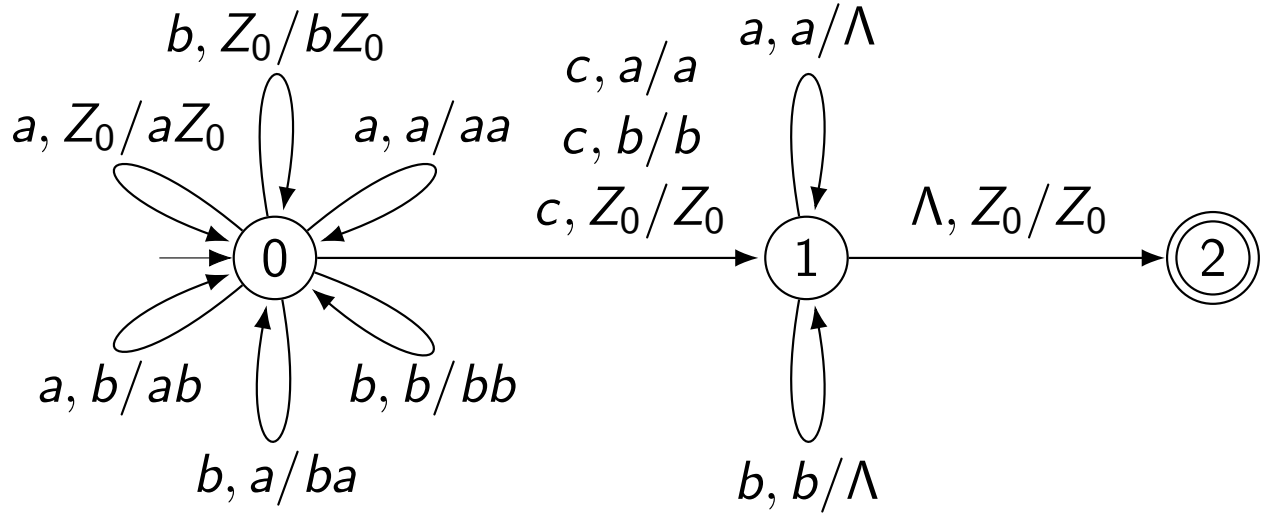


[M] Fig 5.5

notation. . .

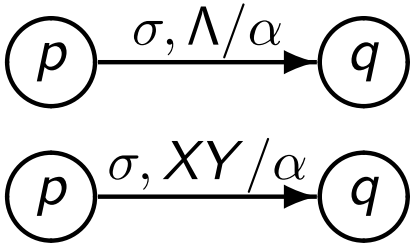


The same PDA twice.
First with some shortcuts in notation.
Second with all individual transitons.



[M] Fig 5.5

Incorrect notations:

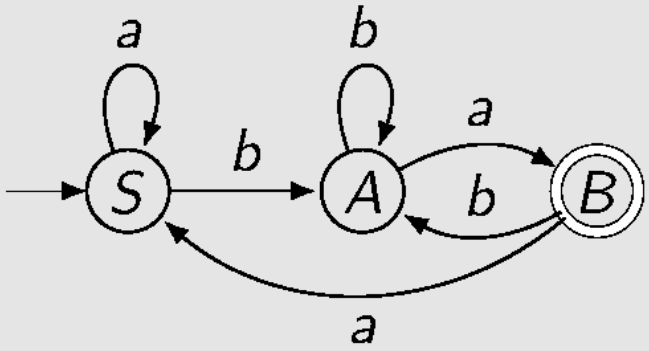


top stack symbol required

remove/consider one stack symbol at a time

From lecture 2:
systematic approach

Example



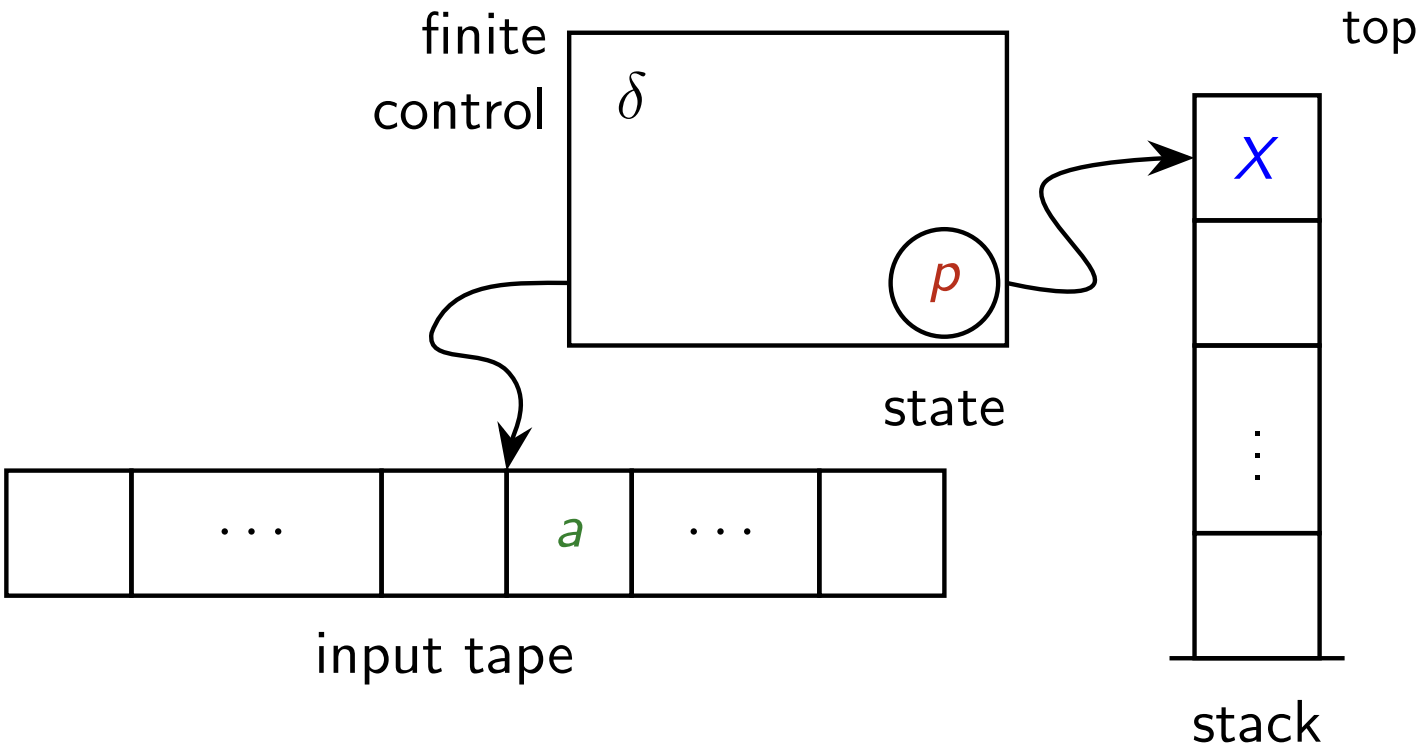
axiom S initial state

$S \to bA \mid aS$ transitions

$A \to bA \mid aB$

$B \to bA \mid aS$

$B \to \Lambda$ accepting state


finite
control
δ
top
X
p
state
a
input tape
stack

From Automaten:

Definition (FA)

[deterministic] finite automaton 5-tuple $M = (Q, \Sigma, q_0, A, \delta)$,

- Q finite set *states*;
- Σ finite *input alphabet*;
- $q_0 \in Q$ *initial* state;
- $A \subseteq Q$ *accepting* states;
- $\delta : Q \times \Sigma \to Q$ *transition function*.

[M] D 2.11 Finite automaton

[L] D 2.1 Deterministic finite accepter, has 'final' states

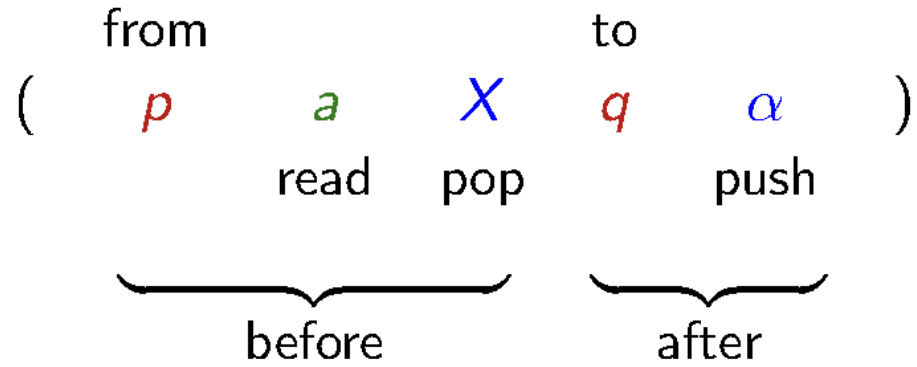
Definition

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



Definition

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

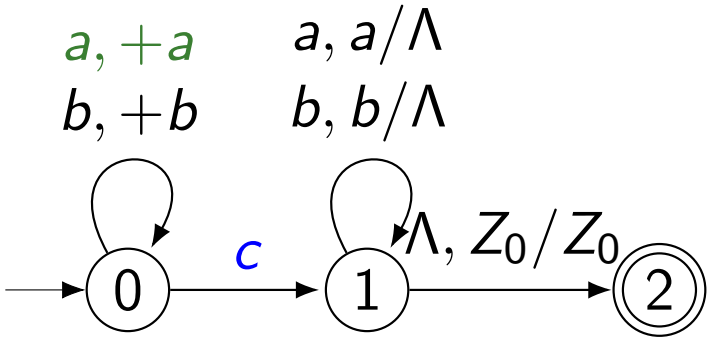
Q	*states*	p, q	
Σ	*input alphabet*	a, b	w, x
Γ	*stack alphabet*	a, b, A, B	α
$q_0 \in Q$	*initial state*		
$Z_0 \in \Gamma$	*initial stack symbol*		
$A \subseteq Q$	*accepting states*		

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

transition function (finite)

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

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



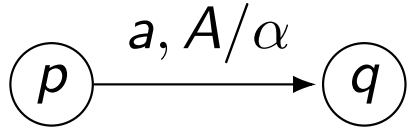
$Q = \{0, 1, 2\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{a, b, Z_0\}$
$q_0 = 0$
$Z_0 = Z_0$
$A = \{2\}$

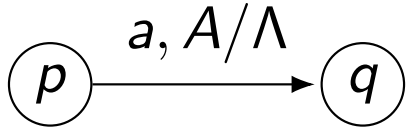
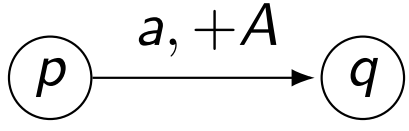
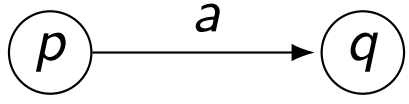
Transition table:

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 $(q, \alpha) \in \delta(p, a, A)$



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

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

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

state, remaining input, stack with top left

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

$\vdash_M^n$ $\vdash_M^*$ $\vdash$ $\vdash^n$ $\vdash^*$

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

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

state, remaining input, stack with top left

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

$\vdash_M^n$ $\quad \vdash_M^*$ $\quad \vdash$ $\quad \vdash^n$ $\quad \vdash^*$

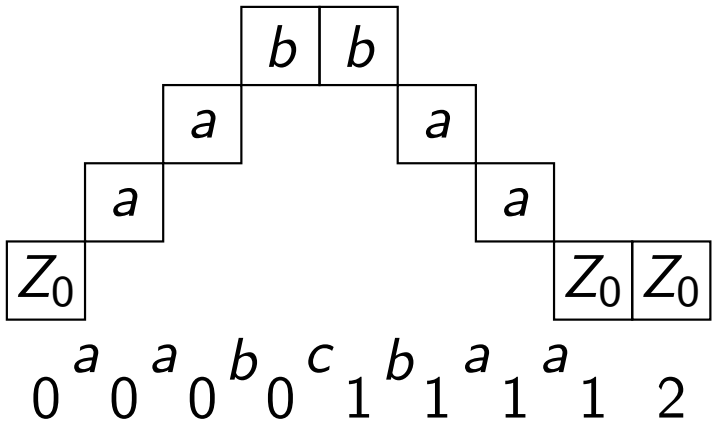
Definition

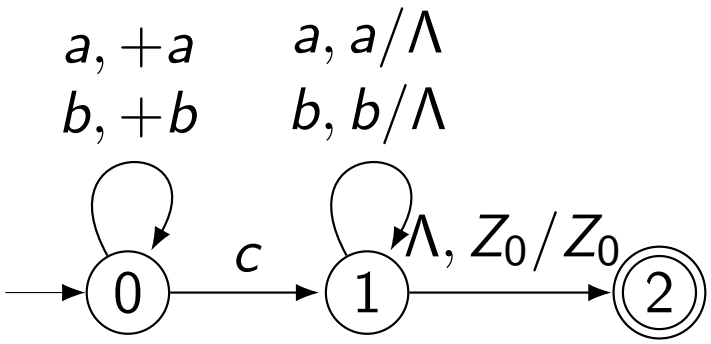
String x accepted by M (by *final state*), if
$(q_0, x, Z_0) \vdash^* (q, \Lambda, \alpha)$ for some $q \in A$, and some $\alpha \in \Gamma^*$
Language accepted by M (by *final state*)
$L(M) = \{ x \in \Sigma^* \mid x \text{ accepted by } M \}$

read complete input, end in accepting state, some path

[M] D 5.2

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

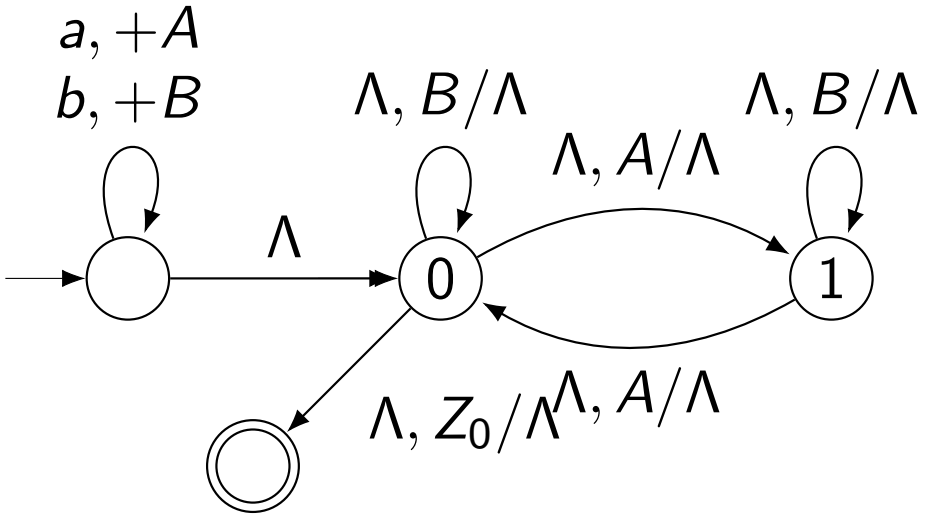


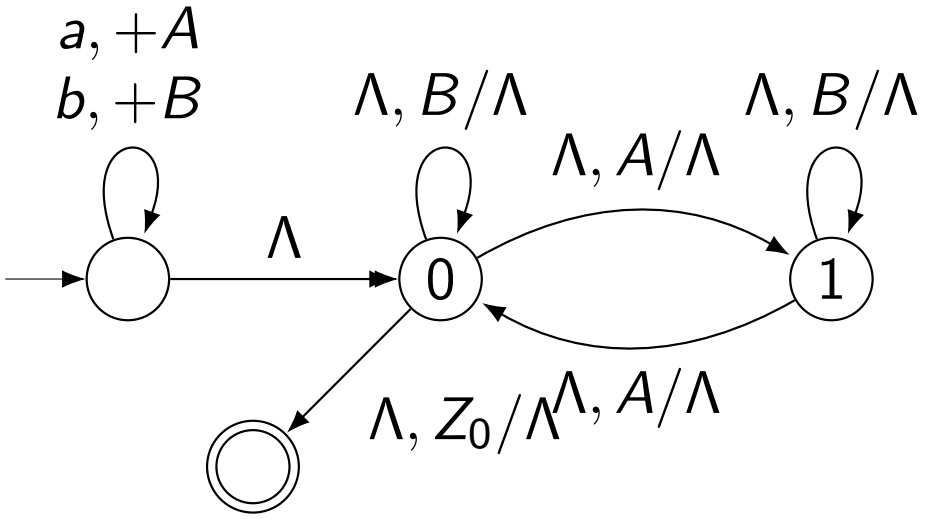


$$
\begin{aligned}
&(0, aabcbaa, Z_0) \vdash \\
&(0, abcbaa, aZ_0) \vdash \\
&(0, bcbaa, aaZ_0) \vdash \\
&(0, cbaa, baaZ_0) \vdash \\
&(1, baa, baaZ_0) \vdash \\
&(1, aa, aaZ_0) \vdash \\
&(1, a, aZ_0) \vdash \\
&(1, \Lambda, Z_0) \vdash \\
&(2, \Lambda, Z_0)
\end{aligned}
$$

[M] Fig 5.5

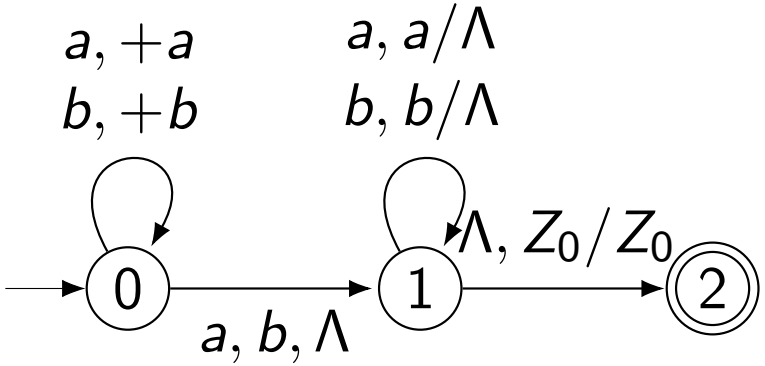
Λ computations





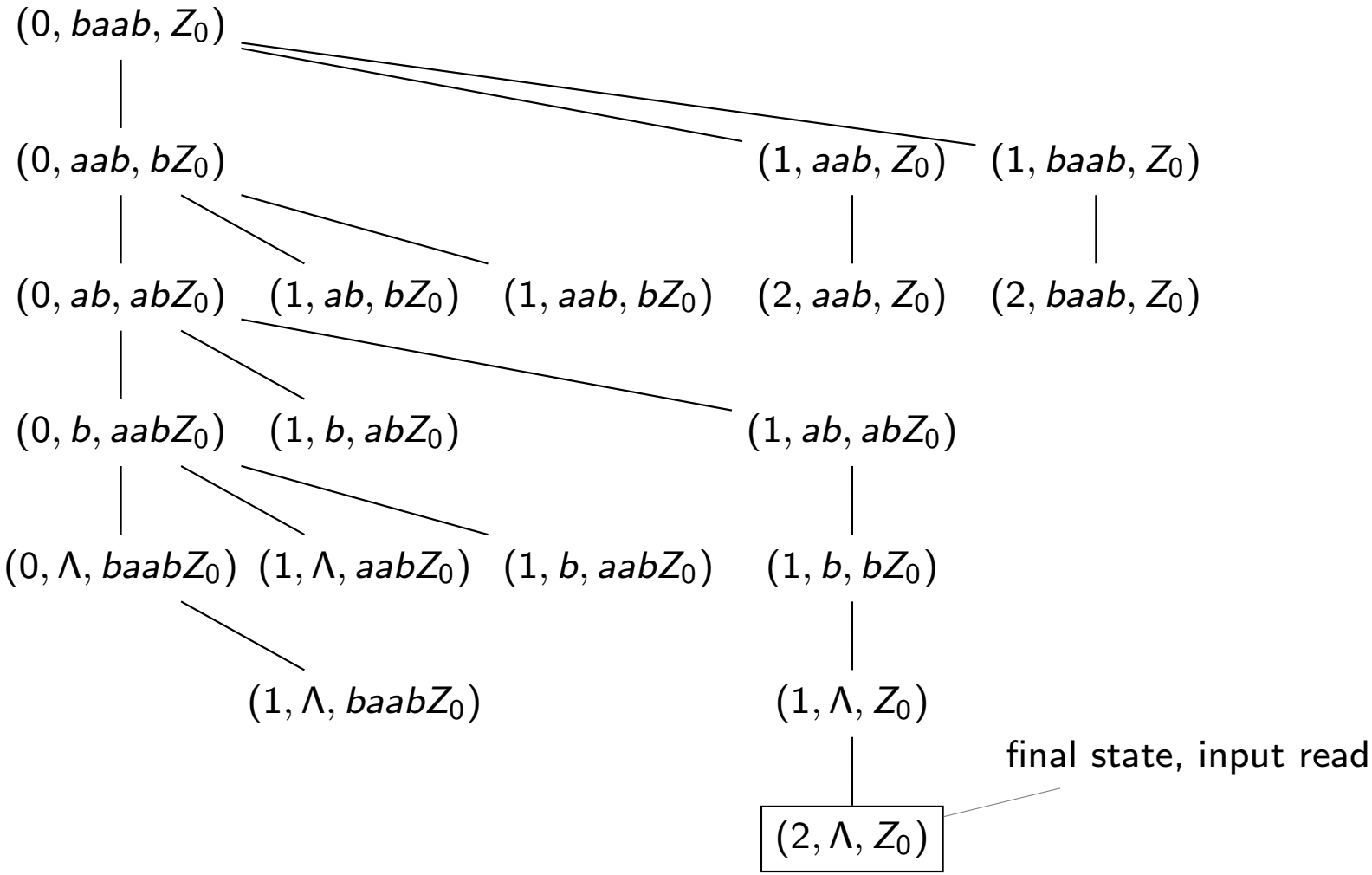
Λ-computations can be very long in PDA, they can even loop.
In the example the input is read and stored on the stack, and at the end of the input it is verified that the string contains an even number of a's.

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



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

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 configuration.

intuitively: no choice for what to do on next input

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

Definition

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

[M] Def 5.10

DPDA $\approx$ DCFL = class of deterministic context-free languages

DPDA for *Balanced*

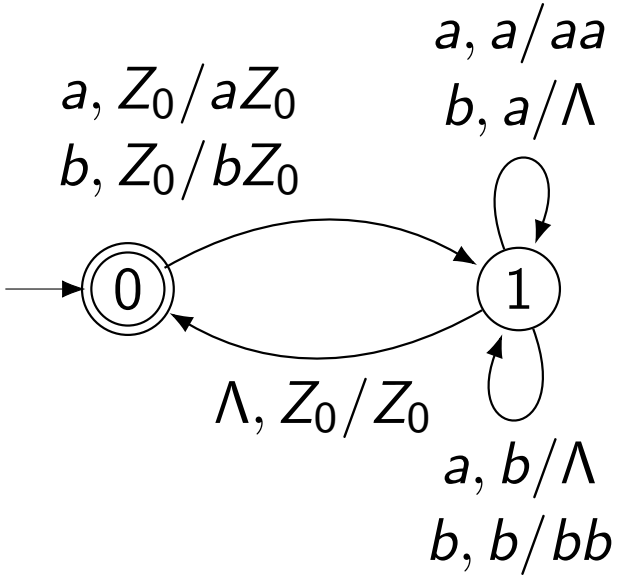
$Balanced = \{\text{balanced strings of brackets [and]}\}$

[M] E 5.11

DPDA for AEqB

[M] E 5.13

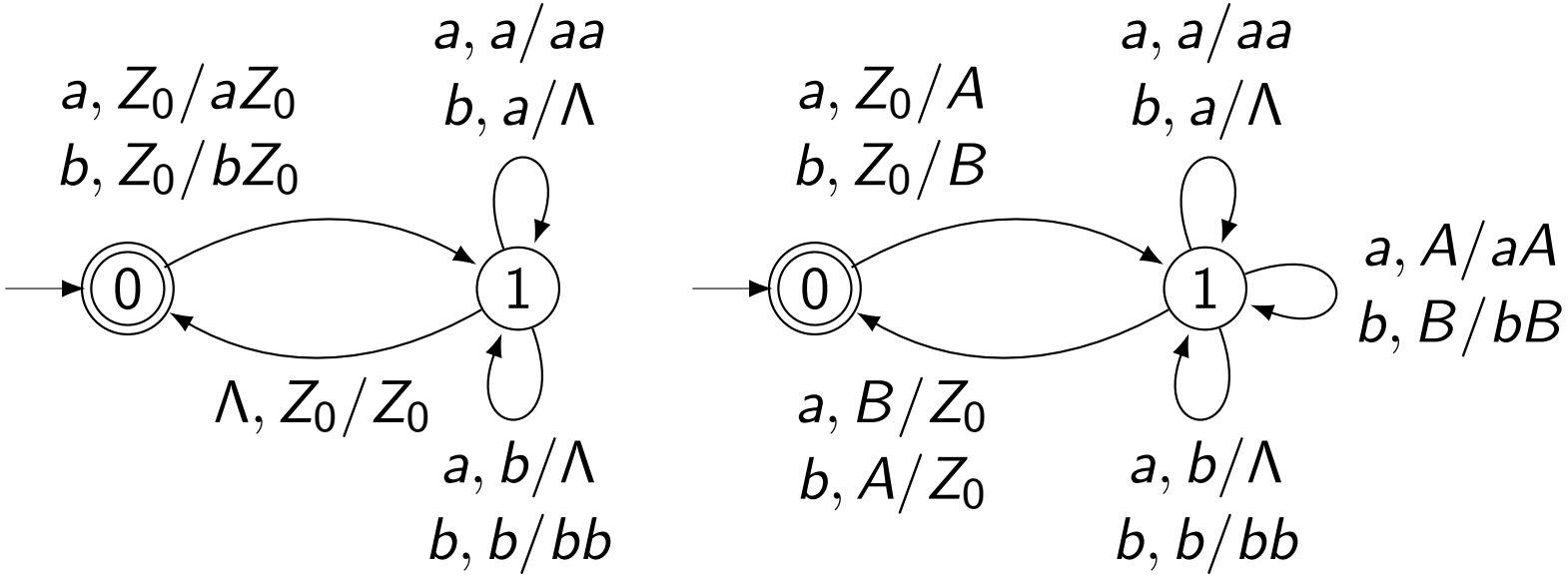
DPDA for AEqB



Without Λ-transitions...

[M] E 5.13

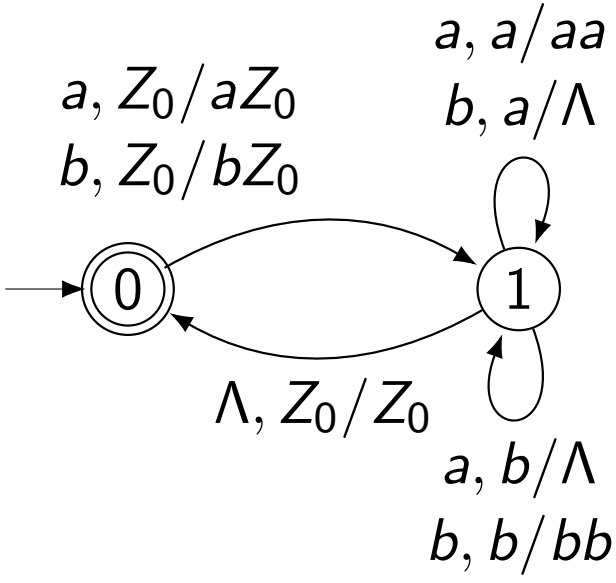
DPDA for AEqB

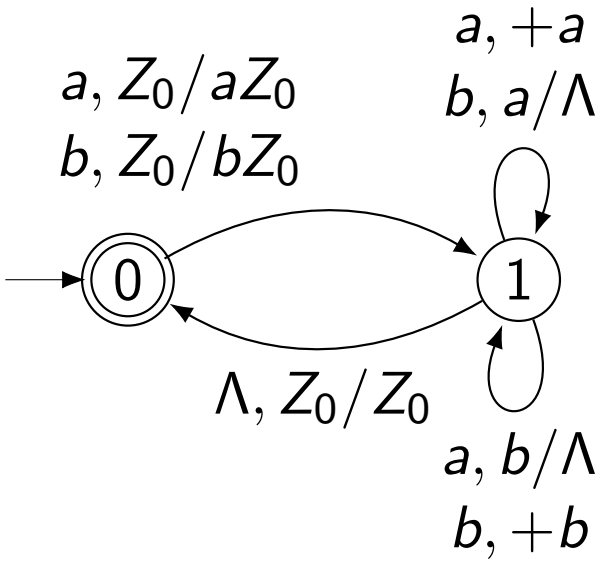


Some nondeterminism...

[M] E 5.13

PDA for AEqB





Computations for $x = abbaab\ldots$

[M] E 5.13