

From lecture 7:

$$A_{eqB} = \{ x \in \{a, b\}^* \mid n_a(x) = n_b(x) \}$$

aaabbb, ababab, aababb, ...

[M] E 4.8

From exercise class 9:

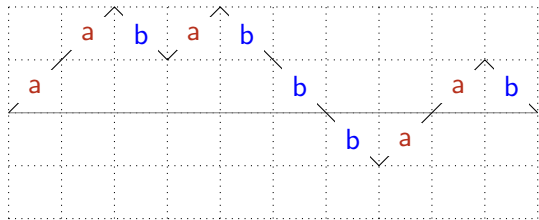
Exercise.

Let G be a context-free grammar with start variable S and the following productions:

$$S \rightarrow aSbS \mid bSaS \mid \Lambda$$

- a. Show that $L(G) = AEqB = \{x \in \{a, b\}^* \mid n_a(x) = n_b(x)\}$. That is, argue why $L(G) \subseteq AEqB$ and why $AEqB \subseteq L(G)$. You do not have to give formal proofs.
- b. Show that G is ambiguous, by giving a string $x \in L(G)$ and two different derivation trees for x in G .
- c. Give an unambiguous context-free grammar for $AEqB$.

$$AeqB = \{ x \in \{a, b\}^* \mid n_a(x) = n_b(x) \}$$



$$AeqB = \{ x \in \{a, b\}^* \mid n_a(x) = n_b(x) \}$$

aaabbb, ababab, aababb, ...

$$S \rightarrow \Lambda \mid aB \mid bA$$

$$A \rightarrow aS \mid bAA$$

$$B \rightarrow bS \mid aBB$$

A generates $n_a(x) = n_b(x) + 1$

B generates $n_a(x) + 1 = n_b(x)$

$S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabSB \Rightarrow \dots$ (different options)

(1) $aabB \Rightarrow aabaBB \Rightarrow aababSB \Rightarrow aababB \Rightarrow aababbS \Rightarrow aababb$

(2) ... (ambiguous, later)

[M] E 4.8

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 10:

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

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 $\text{even}(L)$

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

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 $\text{even}(L)$

variables: $V' = \{X_e, X_o \mid X \in V\}$

axiom: $S' = S_e$

productions: – for every $A \rightarrow BC$ in P we have in P' :

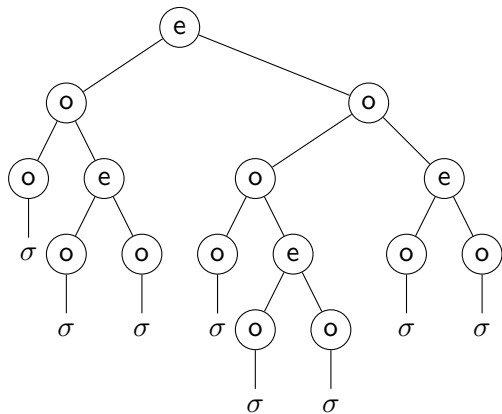
$$A_e \rightarrow B_e C_e \mid B_o C_o \qquad A_o \rightarrow B_e C_o \mid B_o C_e$$

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

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 .



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

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

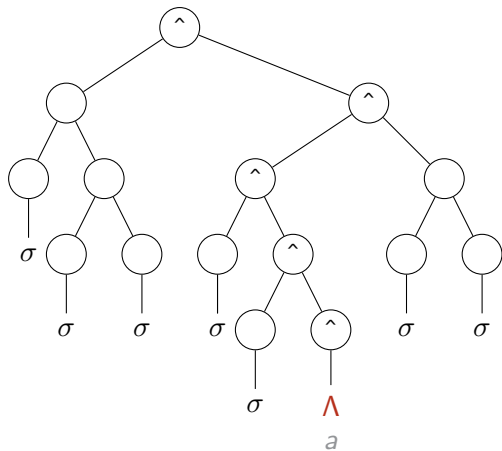
variables: $V' = V \cup \{\hat{X} \mid X \in V\}$

axiom: $S' = \hat{S}$

productions: keep all productions from P , and

– for every $A \rightarrow BC$ add $\hat{A} \rightarrow \hat{B}C \mid B\hat{C}$

– for every $A \rightarrow a$ add $\hat{A} \rightarrow \Lambda$



$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid int$$

$$E \rightarrow E_1 + T_1 \quad E.val = E_1.val + T_1.val$$

$$E \rightarrow T_1 \quad E.val = T_1.val$$

$$T \rightarrow T_1 * F_1 \quad T.val = T_1.val \cdot F_1.val$$

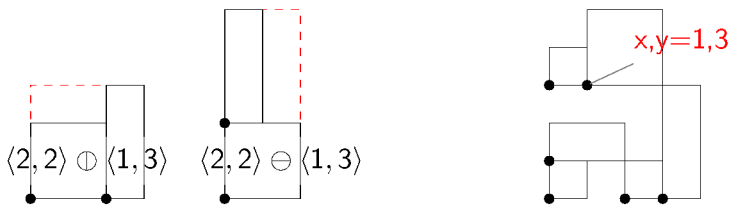
$$T \rightarrow F_1 \quad T.val = F_1.val$$

$$F \rightarrow (E_1) \quad F.val = E_1.val$$

$$F \rightarrow int \quad F.val = IntVal(int)$$

D.E. Knuth. Semantics of Context-Free Languages.

Math. Systems Theory (1968) 127–145 doi:[10.1007/BF01692511](https://doi.org/10.1007/BF01692511)



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

production

$$R \rightarrow \langle E_1, E_2 \rangle$$

$$R \rightarrow (R_1 \oplus R_2)$$

$$R \rightarrow (R_1 \ominus R_2)$$

semantic rule

$$R.b = E_1.val \quad R.h = E_2.val$$

$$R.b = R_1.b + R_2.b$$

$$R.h = \max\{R_1.h, R_2.h\}$$

$$R_1.x = R.x \quad R_2.x = R.x + R_1.b$$

$$R_1.y = R.y \quad R_2.y = R.y$$

$$R.b = \max\{R_1.b, R_2.b\}$$

$$R.h = R_1.h + R_2.h$$

$$R_1.x = R.x \quad R_2.x = R.x$$

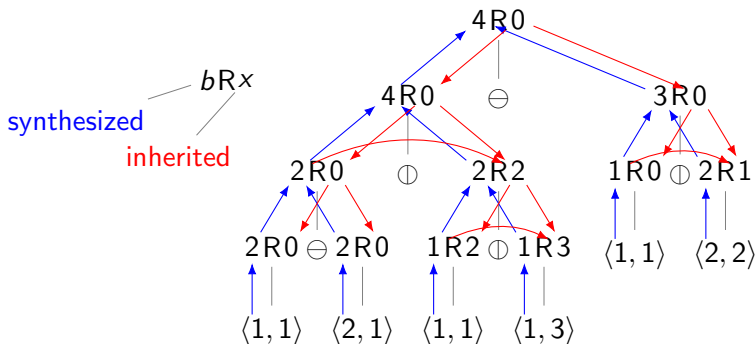
$$R_1.y = R.y \quad R_2.y = R.y + R_1.h$$

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

$$R_1.x = R.x \quad R_2.x = R.x + R_1.b$$

$$R \rightarrow (R_1 \ominus R_2) \quad R.b = \max\{R_1.b, R_2.b\}$$

$$R_1.x = R.x \quad R_2.x = R.x$$



Section 4

Pushdown Automata

- 4 Pushdown Automata
 - Deterministic PDA
 - Empty stack acceptance
 - From CFG to PDA
 - From PDA to CFG

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 $x = abcbaa$

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

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

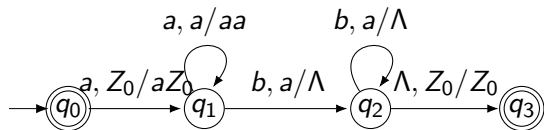
initial q_0, Z_0

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.

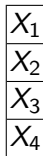
Usual stack operations: pop, top, push

Extra possibility: 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

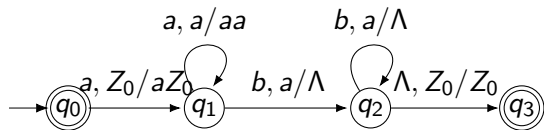
$\alpha = \beta X$ push*

$\alpha = \dots$

Top element X is required to do a move!

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

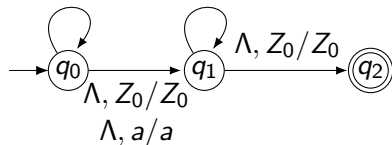
initial q_0, Z_0 , accept $A = \{q_0, q_3\}$



[M] E 5.3

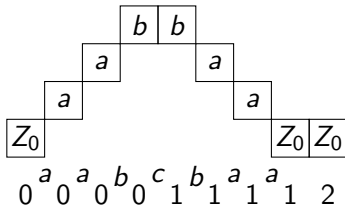
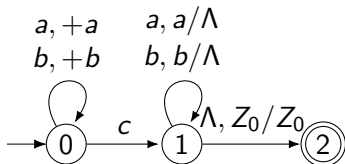
$a, Z_0/aZ_0$

$a, a/aa$ $b, a/\Lambda$



SimplePal =

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

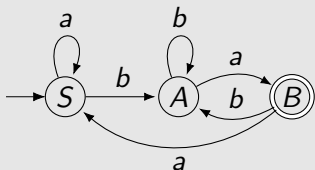


[M] Fig 5.5

notation...

From lecture 8:
systematic approach

Example



axiom S

$S \rightarrow bA \mid aS$

$A \rightarrow bA \mid aB$

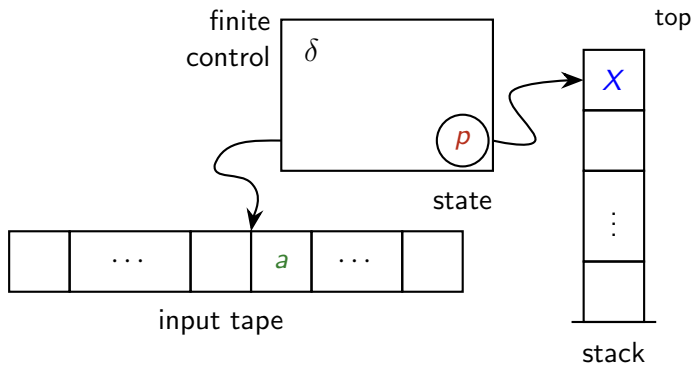
$B \rightarrow bA \mid aS$

$B \rightarrow \Lambda$

initial state

transitions

accepting state



From lecture 2:

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*;
- $\delta : Q \times \Sigma \rightarrow Q$ *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	<i>states</i>	p, q	
Σ	<i>input alphabet</i>	a, b	w, x
Γ	<i>stack alphabet</i>	a, b, A, B	α

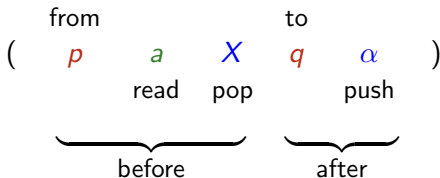
$q_0 \in Q$ *initial state*

$Z_0 \in \Gamma$ *initial stack symbol*

$A \subseteq Q$ *accepting states*

$\delta : \dots \rightarrow \dots$

transition function



Definition

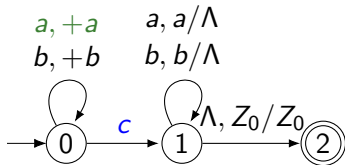
PDA 7-tuple $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$

Q	<i>states</i>	p, q	
Σ	<i>input alphabet</i>	a, b	w, x
Γ	<i>stack alphabet</i>	a, b, A, B	α
$q_0 \in Q$	<i>initial state</i>		
$Z_0 \in \Gamma$	<i>initial stack symbol</i>		
$A \subseteq Q$	<i>accepting states</i>		
$\delta : Q \times (\Sigma \cup \{\Lambda\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$	<i>transition function</i>		(finite)

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

SimplePal =

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



$Q = \{0, 1, 2\}$

$\Sigma = \{a, b, c\}$

$\Gamma = \{a, b, Z_0\}$

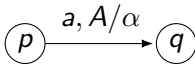
$q_0 = 0$

$Z_0 = Z_0$

$A = \{2\}$

Transition table:

State	Input	Stack Symbol	Move(s)
p	σ	X	$\delta(p, \sigma, X)$
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

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

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

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

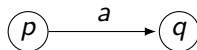
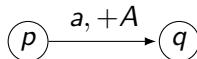
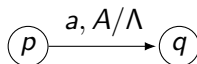
intuitive formalized as

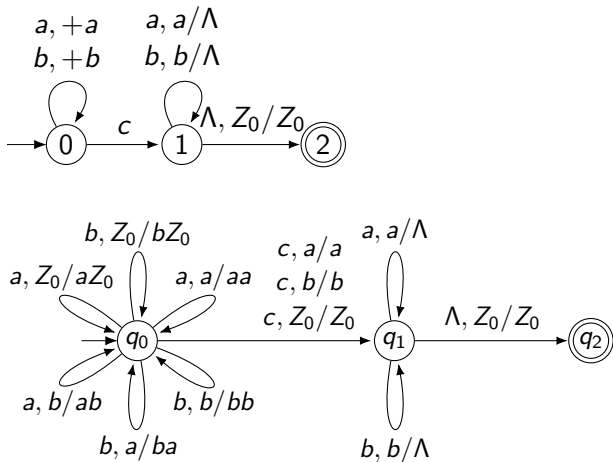
pop A $(q, \Lambda) \in \delta(p, a, A)$ $\alpha = \Lambda$

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 $X \in \Gamma$

convention





[M] Fig 5.5

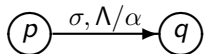
ABOVE

The same PDA twice.

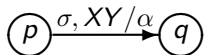
First our version, where we allow some shortcuts in notation.

Second as depicted in the book.

Incorrect notations:



top stack symbol required



remove/consider one stack symbol at a time

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

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

state, remaining input, stack with top left

$$\begin{array}{ccccc} \text{step } (p, ax, B\alpha) \vdash_M (q, x, \beta\alpha) & \text{when } (q, \beta) \in \delta(p, a, B) \\ \vdash_M^n & \vdash_M^* & \vdash & \vdash^n & \vdash^* \end{array}$$

Definition

String x accepted by M (by *final state*), if

$$(q_0, x, Z_0) \vdash^* (q, \Lambda, \alpha) \text{ for some } q \in A, \text{ and some } \alpha \in \Gamma^*$$

Language accepted by M (by *final state*)

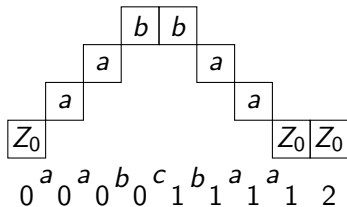
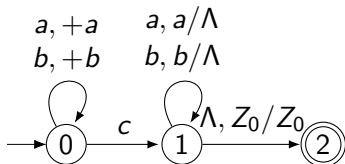
$$L(M) = \{ x \in \Sigma^* \mid x \text{ accepted by } M \}$$

read complete input, end in accepting state, **some path**

[M] D 5.2

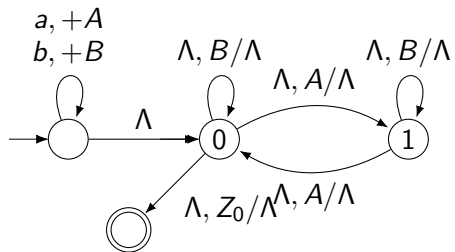
SimplePal =

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



(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, Λ , Z_0) \vdash
 (2, Λ , Z_0)

[M] Fig 5.5

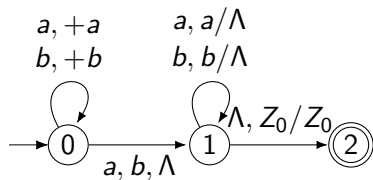


ABOVE

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



$$Q = \{0, 1, 2\}$$

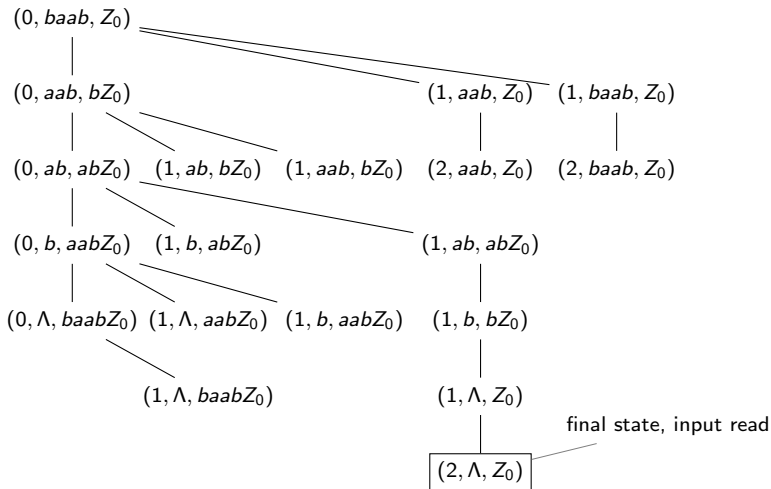
$$\Sigma = \{a, b\}$$

$$\Gamma = \{a, b, Z_0\}$$

$$q_0 = 0$$

$$Z_0 = Z_0$$

$$A = \{2\}$$



[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.

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 $q \in Q, \sigma \in \Sigma, X \in \Gamma$

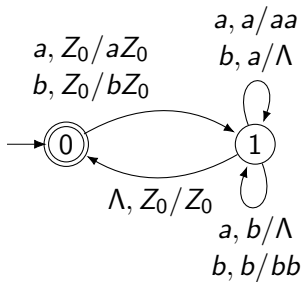
[M] Def 5.10

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

Balanced = {balanced strings of brackets [and]}

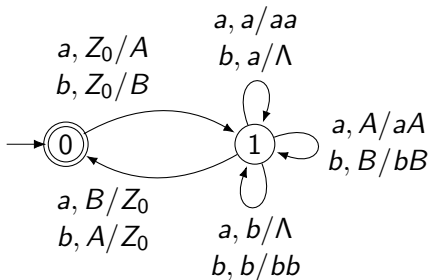
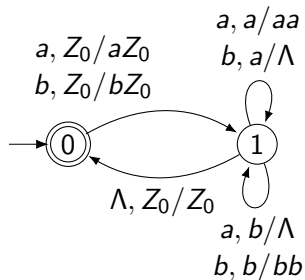
[M] E 5.11

[M] E 5.13



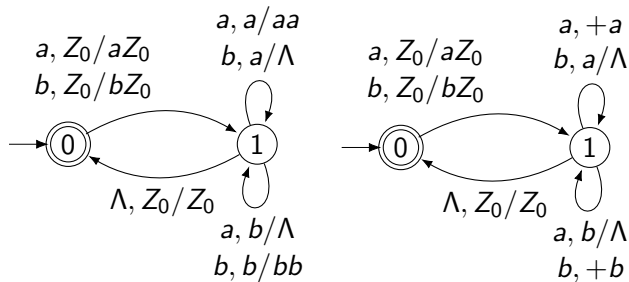
Without Λ -transitions...

[M] E 5.13



Some nondeterminism...

[M] E 5.13



Computations for $x = abbaab \dots$

[M] E 5.13