Section 5

Pushdown Automata

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

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

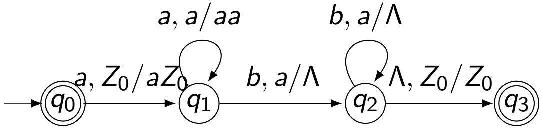
initial q_0, Z_0

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.

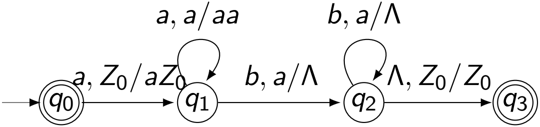
$$\begin{array}{|c|}\hline X_1 \\ \hline X_2 \\ \hline X_3 \\ \hline X_4 \\ \hline \end{array}$$

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

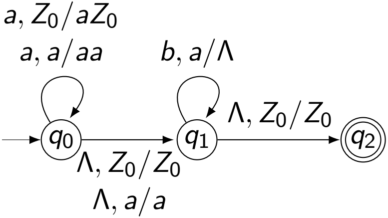
Top element X is required to do a move!

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

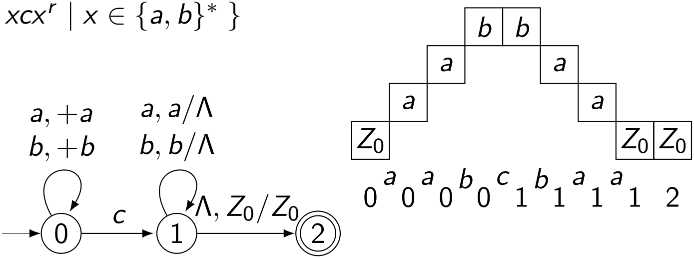
initial q_0, Z_0, accept $A = \{q_0, q_3\}$



[M] E 5.3



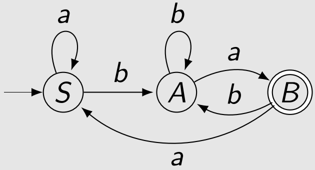
$SimplePal =$
$\{\ xcx^r \mid x \in \{a, b\}^*\ \}$



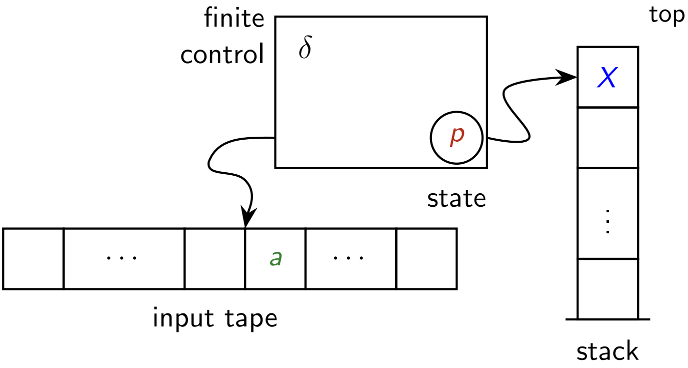
[M] Fig 5.5

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


finite
control
δ
p
state
top
X
$\vdots$
stack
$\cdots$
a
$\cdots$
input tape

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;
- $\delta : Q \times \Sigma \to Q$ *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

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

Definition

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

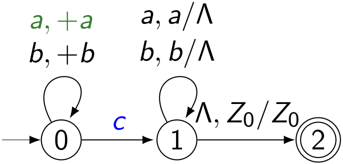
Q	*states*	p, q	
Σ	*input alphabet*	a, b	w, x
Γ	*stack alphabet*	a, b, A, B	α
$q_0 \in Q$	*initial state*		
$Z_0 \in \Gamma$	*initial stack symbol*		
$A \subseteq Q$	*accepting states*		

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

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:

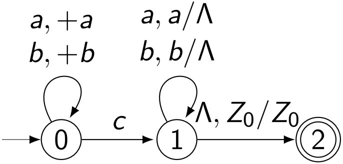
State p	**Input** σ	**Stack Symbol** X	**Move(s)** $\delta(p,\sigma,X)$
0	a	Z_0	$(0, aZ_0)$
0	a	a	$(0, aa)$
0	a	b	$(0, ab)$
0	b	Z_0	$(0, bZ_0)$
0	b	a	$(0, ba)$
0	b	b	$(0, bb)$
0	c	Z_0	$(1, Z_0)$
0	c	a	$(1, a)$
0	c	b	$(1, b)$
1	a	a	$(1, \Lambda)$
1	b	b	$(1, \Lambda)$
1	Λ	Z_0	$(2, Z_0)$
(all other combinations)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

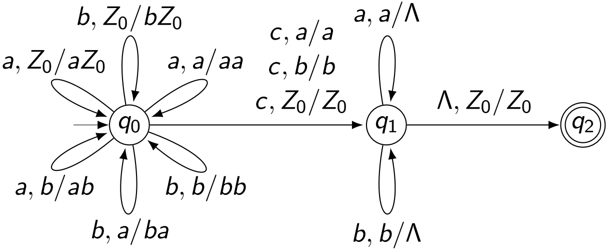
transition $(q, \alpha) \in \delta(p, a, A)$ $\quad (p) \xrightarrow{a, A/\alpha} (q)$

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

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

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 $X \in \Gamma$	$(p) \xrightarrow{a} (q)$


$a, +a$
$b, +b$
$a, a/\Lambda$
$b, b/\Lambda$
c
$\Lambda, Z_0/Z_0$
0
1
2


$b, Z_0/bZ_0$
$a, Z_0/aZ_0$
$a, a/aa$
$a, b/ab$
$b, b/bb$
$b, a/ba$
$c, a/a$
$c, b/b$
$c, Z_0/Z_0$
$a, a/\Lambda$
$b, b/\Lambda$
$\Lambda, Z_0/Z_0$
q_0
q_1
q_2

[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.

Notation

Incorrect notations:

$$(p) \xrightarrow{\sigma, \Lambda/\alpha} (q)$$

top stack symbol required

$$(p) \xrightarrow{\sigma, XY/\alpha} (q)$$

remove/consider one stack symbol at a time

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

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 $(q, \beta) \in \delta(p, a, B)$

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

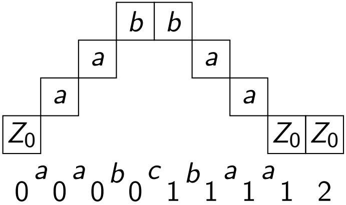
Definition

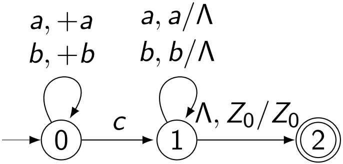
String x accepted by M (by *final state*), if
$(q_0, x, Z_0) \vdash^* (q, \Lambda, \alpha)$ for some $q \in A$, and some $\alpha \in \Gamma^*$
Language accepted by M (by *final state*)
$L(M) = \{\ x \in \Sigma^* \mid x \text{ accepted by } M\ \}$

read complete input, end in accepting state, some path

[M] D 5.2

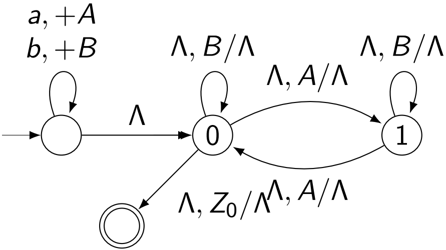
SimplePal $=$
$\{\ xcx^R \mid x \in \{a,b\}^*\ \}$





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

[M] Fig 5.5



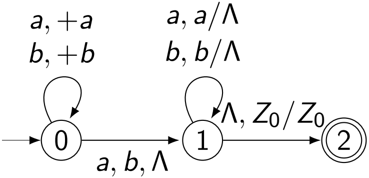
$a, +A$
$b, +B$
$\Lambda, B/\Lambda$
$\Lambda, A/\Lambda$
$\Lambda, B/\Lambda$
Λ
0
1
$\Lambda, Z_0/\Lambda$
$\Lambda, A/\Lambda$

ABOVE

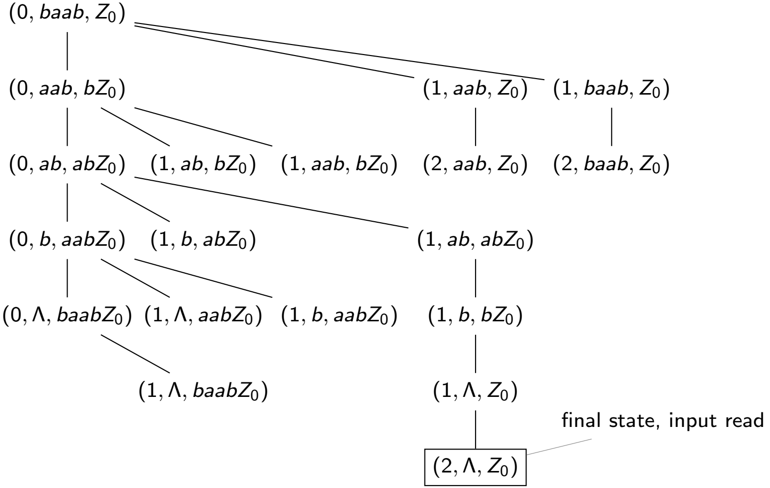
Λ-computations can be very long in PDA, they can even loop.

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



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


$(0, baab, Z_0)$
$(0, aab, bZ_0)$
$(1, aab, Z_0)$
$(1, baab, Z_0)$
$(0, ab, abZ_0)$
$(1, ab, bZ_0)$
$(1, aab, bZ_0)$
$(2, aab, Z_0)$
$(2, baab, Z_0)$
$(0, b, aabZ_0)$
$(1, b, abZ_0)$
$(1, ab, abZ_0)$
$(0, \Lambda, baabZ_0)$
$(1, \Lambda, aabZ_0)$
$(1, b, aabZ_0)$
$(1, b, bZ_0)$
$(1, \Lambda, baabZ_0)$
$(1, \Lambda, Z_0)$
final state, input read
$\boxed{(2, \Lambda, Z_0)}$

[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

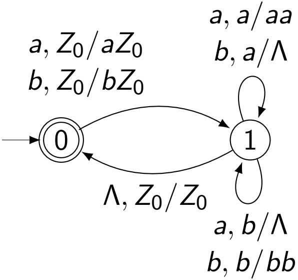
DPDA $\approx$ DCFL $=$ class of deterministic context-free languages

DPDA for *Balanced*

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

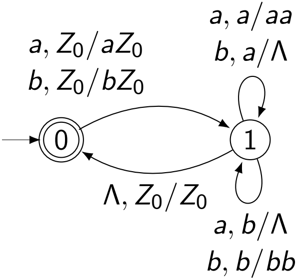
[M] E 5.11

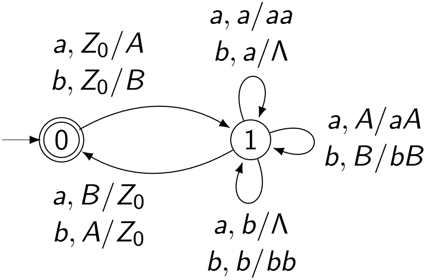
[M] E 5.13



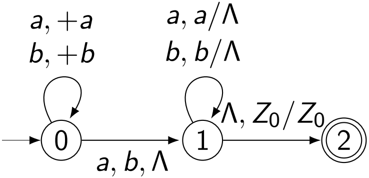
Without Λ-transitions...

[M] E 5.13


$a, Z_0/aZ_0$
$b, Z_0/bZ_0$
$a, a/aa$
$b, a/\Lambda$
0
1
$\Lambda, Z_0/Z_0$
$a, b/\Lambda$
$b, b/bb$


$a, Z_0/A$
$b, Z_0/B$
$a, a/aa$
$b, a/\Lambda$
$a, A/aA$
$b, B/bB$
0
1
$a, B/Z_0$
$b, A/Z_0$
$a, b/\Lambda$
$b, b/bb$

[M] E 5.13



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

Theorem

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

Proof. . .

[M] Thm 5.16

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

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.

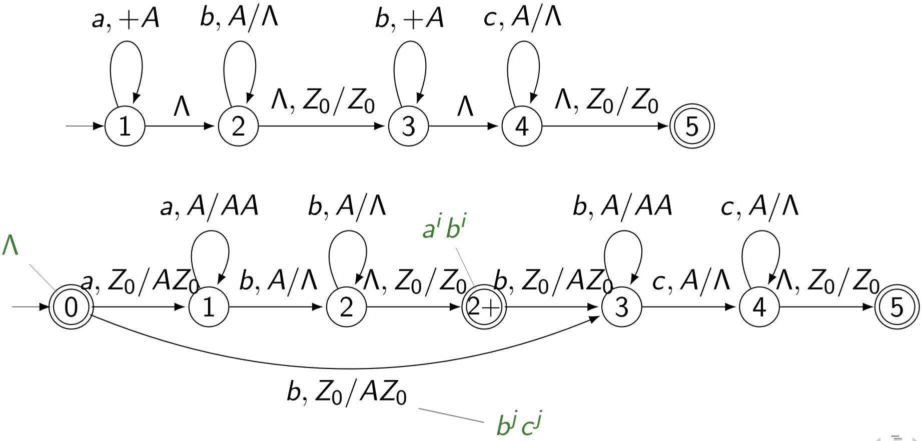
No assumption on form transitions M.

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

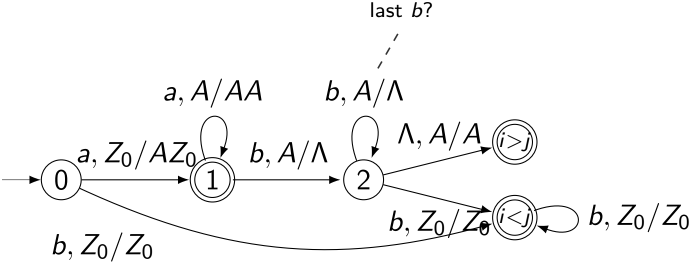
There exist different strings $r, s \in \{a, b\}^*$, such that for every $z \in \{a, b\}^*$, M treats rz and sz the same way.

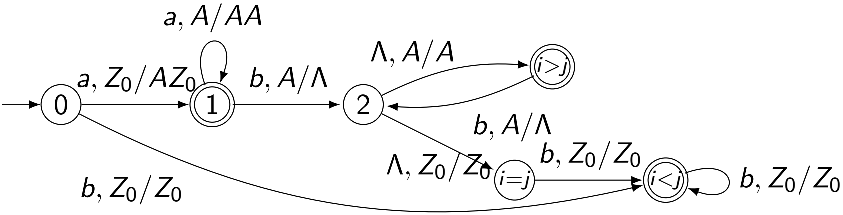
For a string $x \in \{a, b\}^*$, let y_x be a string such that height of stack after xy_x is minimal.
Let α_x be stack after xy_x.
(state, top stack symbol) determines how suffix z is treated.
Infinitely many strings xy_x. Why?
Finitely many pairs (q, X)
Different $r = uy_u$ and $s = vy_v$ arrive at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

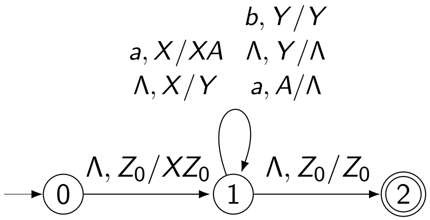
$S \to AB$
$A \to aAb \mid \Lambda$
$B \to bBc \mid \Lambda$

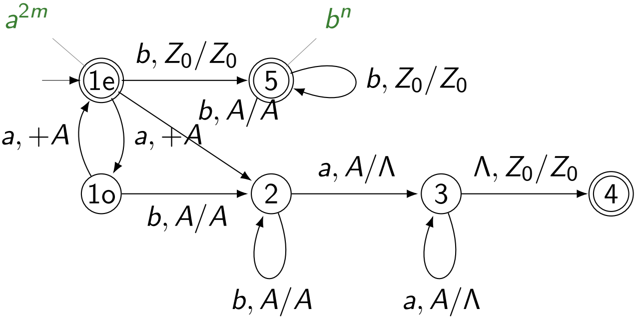


$\{ a^i b^j \mid i \neq j \}$






b, Y/Y
a, X/XA
Λ, Y/Λ
Λ, X/Y
a, A/Λ
Λ, Z0/XZ0
Λ, Z0/Z0
0
1
2


a^2m
b^n
b, Z0/Z0
b, Z0/Z0
b, A/A
a, +A
a, +A
1e
5
1o
2
3
4
a, A/Λ
Λ, Z0/Z0
b, A/A
b, A/A
a, A/Λ

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 $1o$).

$pre(L) = \{\ x\#y \mid x \in L \text{ and } xy \in L\ \}$

$L = Pal = \{\Lambda, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, \ldots\}$
$pre(L) = \ldots$

$L = \{a^i b^j \mid i < j\} = \{b, bb, abb, bbb, abbb, bbbb, aabbb, abbbb, \ldots\}$
$pre(L) = \ldots$

$pre(L) = \{ \, x\#y \mid x \in L \text{ and } xy \in L \, \}$

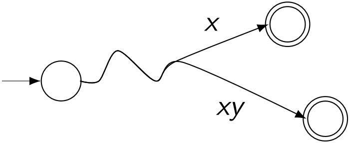
CFL not closed under *pre* $\boxtimes$
DCFL *is* closed under *pre* $\boxtimes$

[M] Exercise 5.20 & 6.22

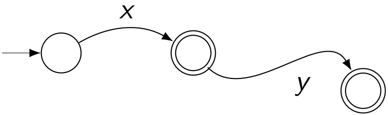
CFL not closed under complement
DCFL is closed under complement $\boxtimes$
(the obvious proof does not work)

CFL is closed under regular operations $\cup, \cdot, *$
DCFL is not closed under either of these $\boxtimes$

language L $\quad x \in L$, $xy \in L$

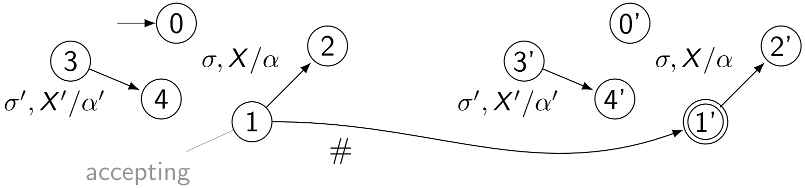


$K = \{\ a^n b^n \mid n \geq 1\ \} \cup \{\ a^n b^m c^n \mid m, n \geq 1\ \}$
$\underline{a^n}\ \underline{b^n} \quad \underline{a^n}\ b^m\ \underline{c^n}$ different behaviour on b's
$pre(K) = \dots$



DCFL is closed under *pre*

$pre(L) = \{\ x\#y \mid x, xy \in L\ \}$

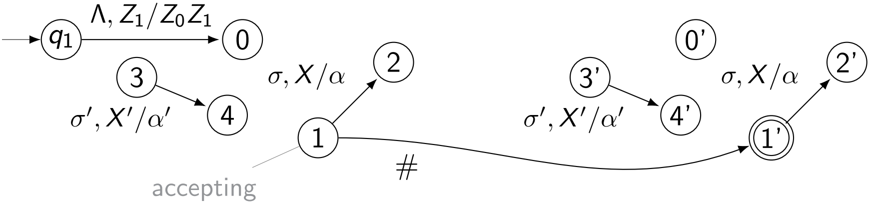


$M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ with $L = L(M)$
construct $M_1 = (Q_1, \Sigma \cup \{\#\}, \Gamma, q_1, Z_1, A_1, \delta_1)$ with $L(M_1) = pre(L)$

- $Q_1 = Q \cup Q'$ where $Q' = \{\ q' \mid q \in Q\ \}$ — primed copy
- $q_1 = q_0$, $Z_1 = Z_0$
- $A_1 = A' = \{\ q' \mid q \in A\ \}$ — accepting states in copy
- $\delta_1(p', \sigma, X) = \{(q', \alpha) \mid (q, \alpha) \in \delta(p, \sigma, X)\}$ — two copies
  for all $p \in A, X \in \Gamma$: $\delta_1(p, \#, X) = \{(p', X)\}$ — move to primed copy

DCFL is closed under *pre*

$pre(L) = \{ \ x\#y \mid x, xy \in L \ \}$



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

construct $M_1 = (Q_1, \Sigma \cup \{\#\}, \Gamma \cup \{Z_1\}, q_1, Z_1, A_1, \delta_1)$ with $L(M_1) = pre(L)$

– $Q_1 = Q \cup Q' \cup \{q_1\}$ where $Q' = \{ \ q' \mid q \in Q \ \}$ primed copy

– $A_1 = A' = \{ \ q' \mid q \in A \ \}$ accepting states in copy

– $\delta_1(p', \sigma, X) = \{(q', \alpha) \mid (q, \alpha) \in \delta(p, \sigma, X)\}$ two copies

$\delta_1(q_1, \Lambda, Z_1) = \{(q_0, Z_0Z_1)\}$ Z_1 under Z_0

for all $p \in A, X \in \Gamma_1$: $\delta_1(p, \#, X) = \{(p', X)\}$ move to primed copy

☒ABOVE

For $K = \{ a^n b^n \mid n \geq 1 \} \cup \{ a^n b^m c^n \mid m, n \geq 1 \}$
we have $pre(K) = K\# \cup \{a^n b^n \# b^k c^n \mid n \geq 1, k \geq 0\}$.
This language is not context-free, but K is, and thus the context-free languages are not closed under pre.

Again, this construction works because (for deterministic automata) the computation on uv *must* extend the computation on u.

Note the resulting PDA might not be deterministic at accepting states in original Q (like node 1 in the diagram), if that node has an outgoing Λ-transition.
There is however a method that avoids Λ-transitions at accepting states. Whenever $(q, \alpha) \in \delta(p, \Lambda, A)$ for an accepting state p, just 'predict' the next letter σ read, add a new state (q, σ), add $((q, \sigma), \alpha)$ to $\delta(p, \sigma, A)$ (which was empty beforehand, why?). Do this for every σ, and remove the Λ-transition. Then keep simulating Λ-transitions, until σ is read.

Quiz

Homework 3!