Huiswerk

- Huiswerkopgave 2: deze week beschikbaar?

From exercise class 6:

Exercise 5.17.

Show that if L is accepted by a PDA, then L is accepted by a PDA in which every move

$*$ either pops something from the stack (i.e., removes a stack symbol without putting anything else on the stack);

$*$ or pushes a single symbol onto the stack on top of the symbol that was previously on top;

$*$ or leaves the stack unchanged.

Hence, each action on the stack due to a move in the PDA has one of the following forms:

$*$ either X/Λ (with $X \in \Gamma$),

$*$ or X/YX (with $X, Y \in \Gamma$),

$*$ or X/X (with $X \in \Gamma$).

From exercise class 6:

Exercise 5.28.

In each case below, you are given a CFG G and a string x that it generates. Draw the nondeterministic top-down PDA $NT(G)$.

Trace a sequence of moves in $NT(G)$ by which x is accepted, showing at each step the state, the unread input, the stack contents and the action, in a table as follows:

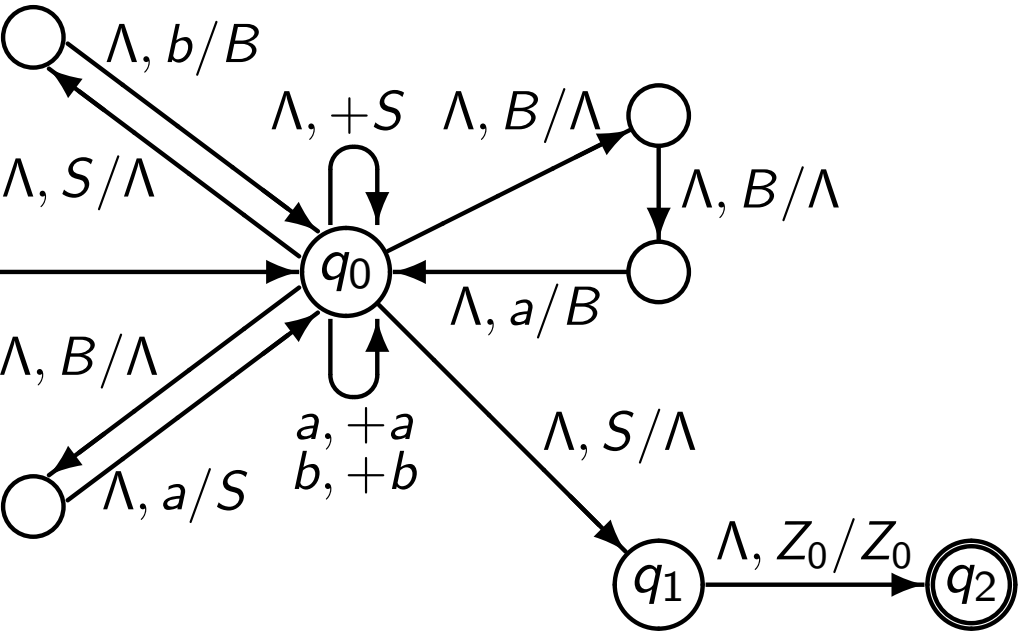
state	unread input	stack contents	action
q_0	$[a * a + a]$	Z_0	
q_1	$[a * a + a]$	$S\ Z_0$	$S \to [S]$
q_1	$[a * a + a]$	$[S]\ Z_0$	match [

Bottom-up = shift-reduce

From lecture 6:

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

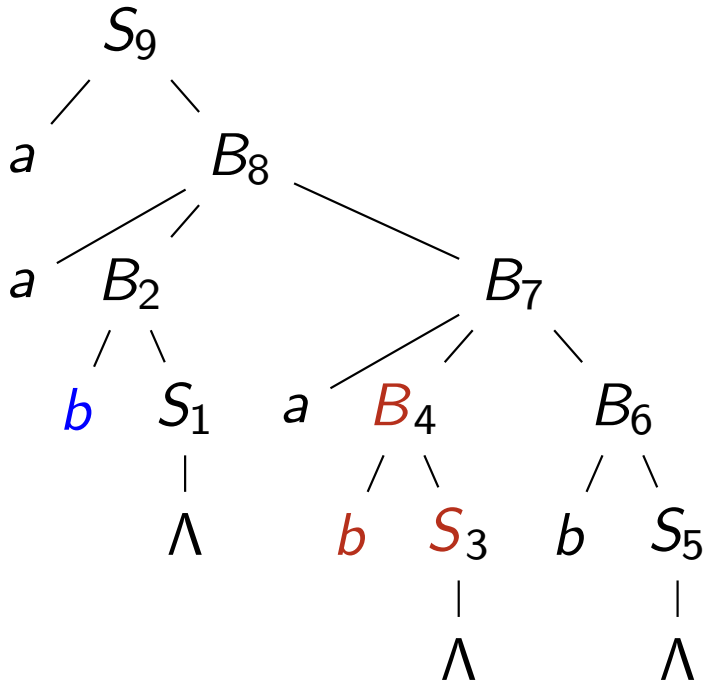
$$S \to \Lambda \mid aB \mid bA$$
$$A \to aS \mid bAA$$
$$B \to bS \mid aBB$$



+ states/transitions for other productions

	stack^r	input	
q_0	Z_0	*aababb*	shift *a*
q_0	Z_0 *a*	*a ababb*	shift *a*
q_0	Z_0 *aa*	*aa babb*	shift *b*
q_0	Z_0 *aab*	*aab abb*	$1 : S \to \Lambda$
q_0	Z_0 *aabS*	*aab abb*	$2 : B \to bS$
q_0	Z_0 *aaB*	*aab abb*	shift *a*
q_0	Z_0 *aaBa*	*aaba bb*	shift *b*
q_0	Z_0 *aaBab*	*aabab b*	$3 : S \to \Lambda$
q_0	Z_0 *aaBabS*	*aabab b*	$4 : B \to bS$
q_0	Z_0 *aaBaB*	*aabab b*	shift *b*
q_0	Z_0 *aaBaBb*	*aababb*	$5 : S \to \Lambda$
q_0	Z_0 *aaBaBbS*	*aababb*	$6 : B \to bS$
q_0	Z_0 *aaBaBB*	*aababb*	$7 : B \to aBB$
q_0	Z_0 *aaBB*	*aababb*	$8 : B \to aBB$
q_0	Z_0 *aB*	*aababb*	$9 : S \to aB$
q_0	Z_0 *S*	*aababb*	
q_1	Z_0	*aababb*	
q_2	Z_0	*aababb*	

Bottom-up = shift-reduce

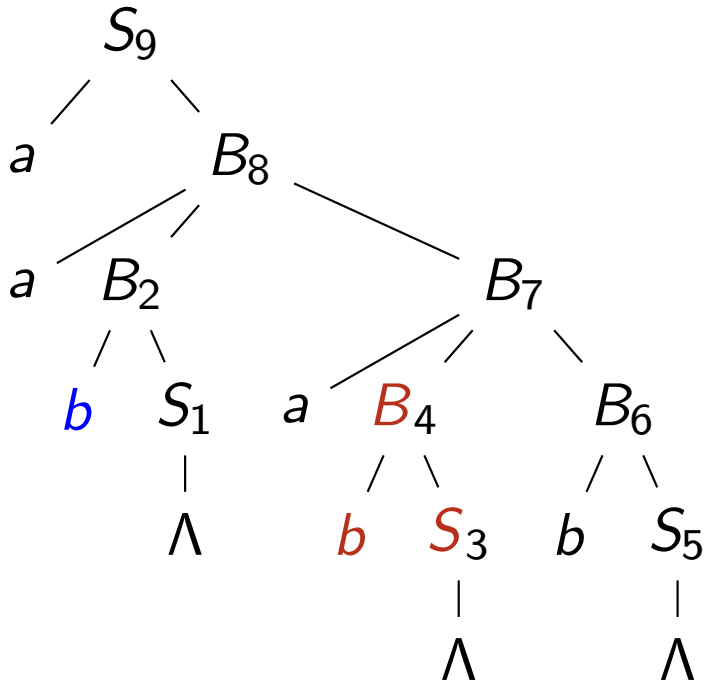


postorder: rightmost

$S \overset{r}{\Rightarrow}_9 aB \Rightarrow_8 aaBB \Rightarrow_7 aaBaBB \Rightarrow_6$
$aaBaBbS \Rightarrow_5 aaBaBb \Rightarrow_4 aaBabSb$
$\Rightarrow_3 aaBabb \Rightarrow_2 aabSabb \Rightarrow_1 aababb$

	stack^r	input	
q_0	Z_0	$aababb$	shift a
q_0	$Z_0\ a$	$a\ ababb$	shift a
q_0	$Z_0\ aa$	$aa\ babb$	shift b
q_0	$Z_0\ aab$	$aab\ abb$	$1 : S \to \Lambda$
q_0	$Z_0\ aabS$	$aab\ abb$	$2 : B \to bS$
q_0	$Z_0\ aaB$	$aab\ abb$	shift a
q_0	$Z_0\ aaBa$	$aaba\ bb$	shift b
q_0	$Z_0\ aaBab$	$aabab\ b$	$3 : S \to \Lambda$
q_0	$Z_0\ aaBabS$	$aabab\ b$	$4 : B \to bS$
q_0	$Z_0\ aaBaB$	$aabab\ b$	shift b
q_0	$Z_0\ aaBaBb$	$aababb$	$5 : S \to \Lambda$
q_0	$Z_0\ aaBaBbS$	$aababb$	$6 : B \to bS$
q_0	$Z_0\ aaBaBB$	$aababb$	$7 : B \to aBB$
q_0	$Z_0\ aaBB$	$aababb$	$8 : B \to aBB$
q_0	$Z_0\ aB$	$aababb$	$9 : S \to aB$
q_0	$Z_0\ S$	$aababb$	
q_1	Z_0	$aababb$	
q_2	Z_0	$aababb$	

Bottom-up = shift-reduce



postorder: rightmost, in reverse

$S \overset{r}{\Rightarrow}_9 aB \Rightarrow_8 aaBB \Rightarrow_7 aaBaBB \Rightarrow_6$
$aaBaBbS \Rightarrow_5 aaBaBb \Rightarrow_4 aaBabSb$
$\Rightarrow_3 aaBabb \Rightarrow_2 aabSabb \Rightarrow_1 aababb$

	stack^r	input	
q_0	Z_0	$aababb$	shift a
q_0	$Z_0\ a$	$a\ ababb$	shift a
q_0	$Z_0\ aa$	$aa\ babb$	shift b
q_0	$Z_0\ aab$	$aab\ abb$	$1 : S \to \Lambda$
q_0	$Z_0\ aabS$	$aab\ abb$	$2 : B \to bS$
q_0	$Z_0\ aaB$	$aab\ abb$	shift a
q_0	$Z_0\ aaBa$	$aaba\ bb$	shift b
q_0	$Z_0\ aaBab$	$aabab\ b$	$3 : S \to \Lambda$
q_0	$Z_0\ aaBabS$	$aabab\ b$	$4 : B \to bS$
q_0	$Z_0\ aaBaB$	$aabab\ b$	shift b
q_0	$Z_0\ aaBaBb$	$aababb$	$5 : S \to \Lambda$
q_0	$Z_0\ aaBaBbS$	$aababb$	$6 : B \to bS$
q_0	$Z_0\ aaBaBB$	$aababb$	$7 : B \to aBB$
q_0	$Z_0\ aaBB$	$aababb$	$8 : B \to aBB$
q_0	$Z_0\ aB$	$aababb$	$9 : S \to aB$
q_0	$Z_0\ S$	$aababb$	
q_1	Z_0	$aababb$	
q_2	Z_0	$aababb$	

ABOVE

To write down the construction of the shift-reduce PDA for a given CFG, we have two technical problems.

Consider a production $A \to \alpha$

First the stack (in standard notation) now contains the string α in reverse.

Second, we pop α, that is, several symbols, rather than exactly one. This can be simulated by popping the symbols one-by-one, using $|\alpha|$ separate transitions.

shift $\delta(q_0, \sigma, X) = \{(q_0, \sigma X)\}$ for $\sigma \in \Sigma$, $X \in \Gamma$

reduce '$\delta^*(q_0, \Lambda, \alpha^r) \ni (q_0, A)$' for $A \to \alpha$ in P

Special case: $\alpha = \Lambda$

Definition

The Nondeterministic Bottom-Up PDA $NB(G)$
Let $G = (V, \Sigma, S, P)$ be a context-free grammar.
The nondeterministic bottom-up PDA corresponding to G is
$NB(G) = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$, defined as follows:
Q contains the initial state q_0, the state q_1, and the (only) accepting state q_2, together with other states to be described shortly.
$\Gamma = \ldots$

[M] D 5.22

Definition

The Nondeterministic Bottom-Up PDA $NB(G)$
Let $G = (V, \Sigma, S, P)$ be a context-free grammar.
The nondeterministic bottom-up PDA corresponding to G is $NB(G) = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$, defined as follows:
Q contains the initial state q_0, the state q_1, and the (only) accepting state q_2, together with other states to be described shortly.
$\Gamma = V \cup \Sigma \cup \{Z_0\}$

[M] D 5.22

Definition

The Nondeterministic Bottom-Up PDA $NB(G)$ (continued)
For every $\sigma \in \Sigma$ and every $X \in \Gamma$, $\delta(q_0, \sigma, X) = \{(q_0, \sigma X)\}$. This is a *shift* move.
For every production $B \to \alpha$ in G, and every nonnull string $\beta \in \Gamma^*$,
$(q_0, \Lambda, \alpha^r \beta) \vdash^* (q_0, \Lambda, B\beta)$,
where this *reduction* is a sequence of one or more moves in which, if there is more than one, the intermediate configurations involve other states that are specific to this sequence and appear in no other moves of $NB(G)$.
One of the elements of $\delta(q_0, \Lambda, S)$ is (q_1, Λ),
and $\delta(q_1, \Lambda, Z_0) = \{(q_2, Z_0)\}$.

[M] D 5.22

Theorem

If G is a context-free grammar, then the nondeterministic bottom-up PDA $\mathrm{NB}(G)$ *accepts the language* $L(G)$.

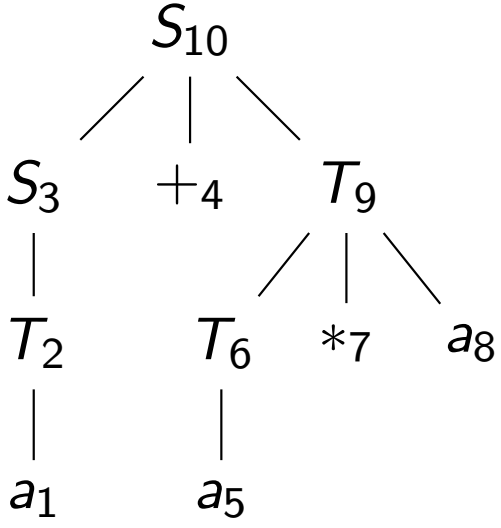
The details of the proof of this result do not have to be known for the exam.

[M] Th 5.23

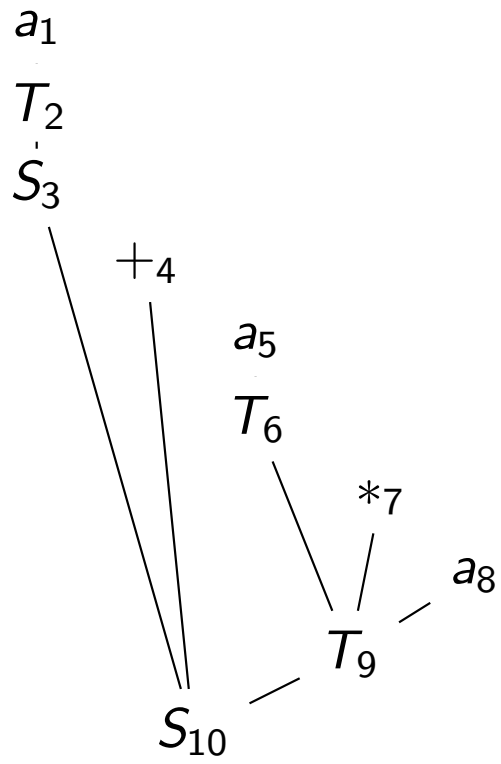
Example: algebraic expressions

shift-reduce

post-order reduction $\equiv$ rightmost derivation, bottom-up



stack [reverse]	input
Z_0	$a + a * a$
$Z_0\ a_1$	$+a * a$
$Z_0\ T_2$	$+a * a$
$Z_0\ S_3$	$+a * a$
$Z_0\ S_3\ +_4$	$a * a$
$Z_0\ S_3\ +_4\ a_5$	$*a$
$Z_0\ S_3\ +_4\ T_6$	$*a$
$Z_0\ S_3\ +_4\ T_6\ *_7$	a
$Z_0\ S_3\ +_4\ T_6\ *_7\ a_8$	
$Z_0\ S_3\ +_4\ T_9$	
$Z_0\ S_{10}$	
$-$	



Due to Chomsky, Evey, and Schützenberger (1962/3).

Theorem

Context-free grammars and Pushdown automata are equivalent.

$\hookrightarrow$(1) PDA acceptance by empty stack

$\hookrightarrow$(2) triplet construction, CFG nonterminals $[p, A, q]$ for PDA computations

References

N. Chomsky. Context-free grammars and push-down storage. Quarterly Progress Report No. 65, Research Laboratory of Electronics, M.I.T., Princeton, New Jersey (1962)

R.J. Evey. The theory and application of pushdown store machines. In Mathematical Linguistics and Automatic Translation, NSF-IO, pages 217–255. Harvard University, May 1963,
and

M. P. Schützenberger. On context-free languages and pushdown automata. Inform. and Control, 6:217–255, 1963. doi:10.1016/S0019-9958(63)90306-1

From lecture 5:

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

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

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

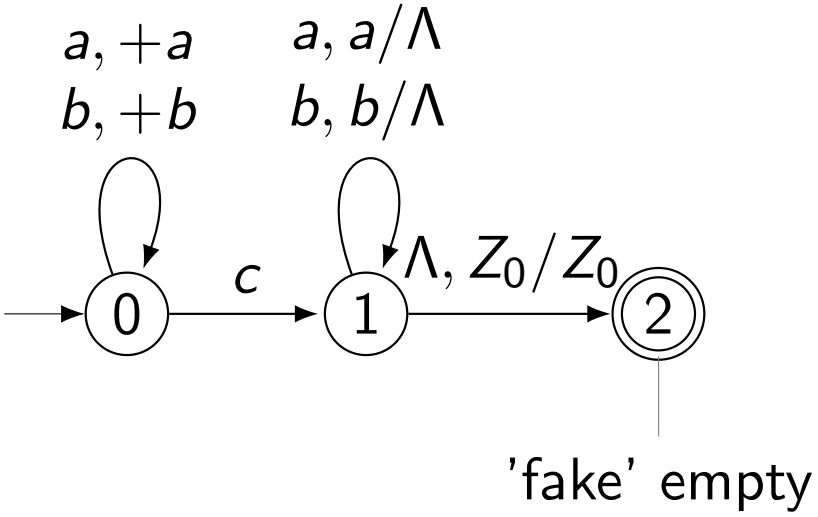
Definition

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

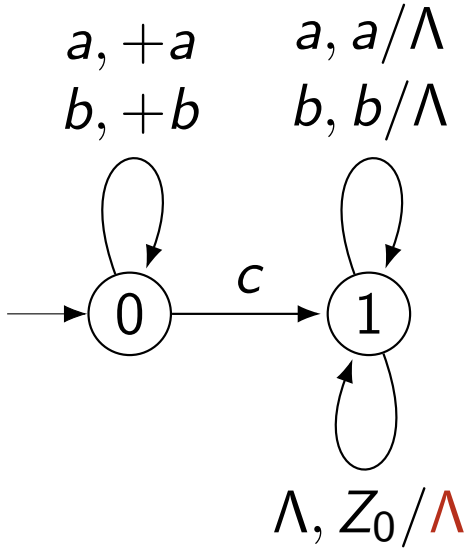
read complete input, end in accepting state, some path

[M] D 5.2

Empty stack

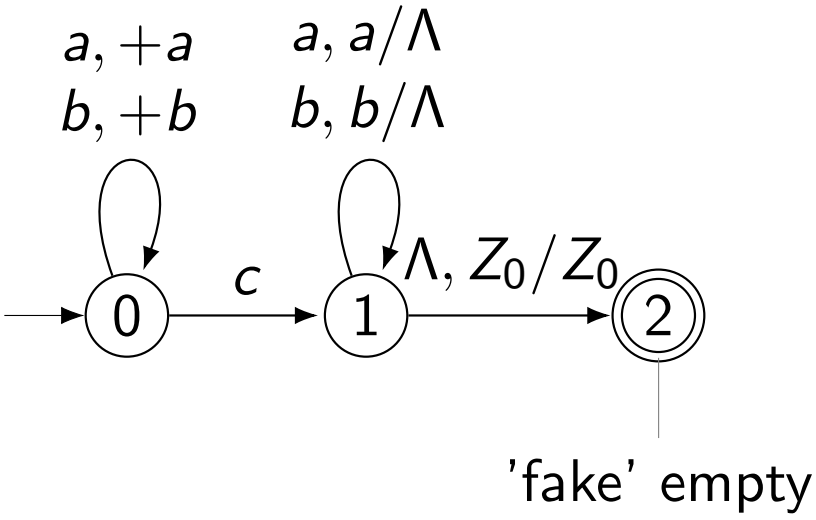


check state

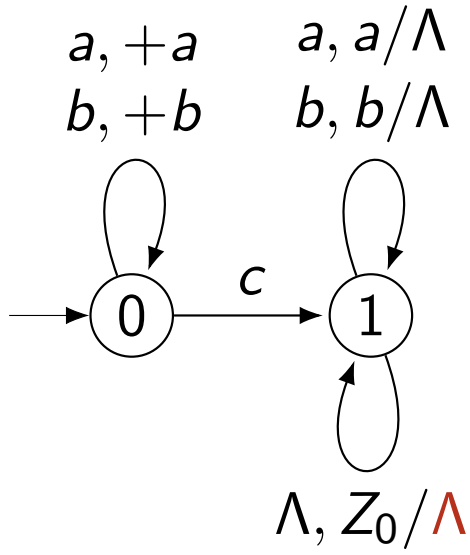


check stack

Empty stack



check state



check stack

On many cases the PDA moves to the accepting state after checking that the stack is empty, when the topmost symbol is a special Z_0 that always has been at the bottom of the stack.
It may be more natural to accept directly by looking at the stack rather than by looking at the state. This leads to the notion of the *empty stack* language of a PDA.

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

Definition

Language accepted by M by *empty stack*
$L_e(M) = \{ x \in \Sigma^* \mid (q_0, x, Z_0) \vdash^* (q, \Lambda, \Lambda) \text{ for some state } q \in Q \}$

[M] D 5.27

Theorem

If M is a PDA then there is a PDA M_1 such that $L_e(M_1) = L(M)$.

Sketch of proof...

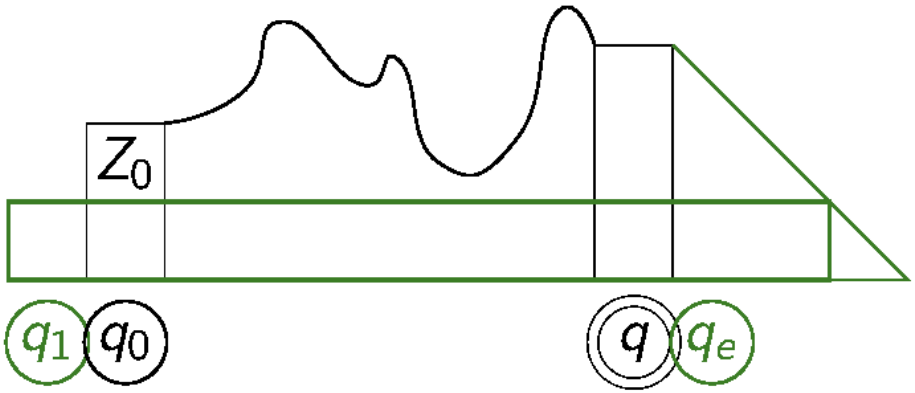
[M] Th 5.28

Final state to empty stack

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

Final state to empty stack

Simulate $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$
– empty stack 'at' final state
– prohibit early empty stack

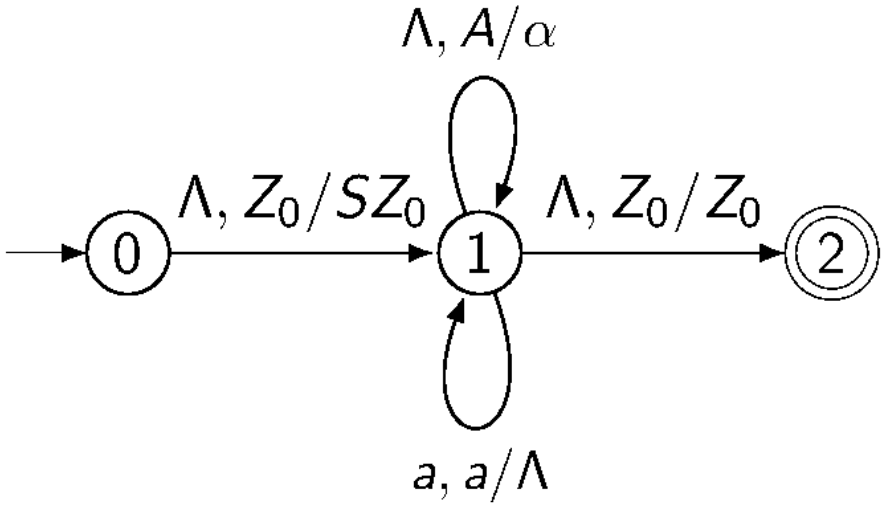


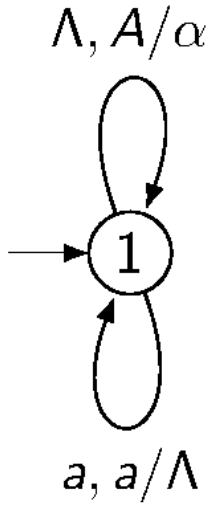
Construction PDA $M_1 = (Q_1, \Sigma, \Gamma_1, q_1, Z_1, A_1, \delta_1)$ such that $L_e(M_1) = L(M)$

– $Q_1 = Q \cup \{q_1, q_e\}$
– $\Gamma_1 = \Gamma \cup \{Z_1\}$
– new instructions:
  $\delta_1(q_1, \Lambda, Z_1) = \{(q_0, Z_0 Z_1)\}$
  $\delta_1(q, \Lambda, X) \ni (q_e, X)$ for $q \in A$, and $X \in \Gamma_1$
  $\delta_1(q_e, \Lambda, X) = \{(q_e, \Lambda)\}$ for $X \in \Gamma_1$

Expand-match with empty stack

$A \to \alpha \in P,\ a \in \Sigma$





Theorem

For every CFL L there exists a single state PDA M such that $L_e(M) = L$.

ABOVE

Now that we have empty stack acceptance we can reconsider the expand-match technique. In fact we do not need two extra states to introduce a bottom of stack symbol, and can make a single state PDA.

BELOW

The expand-match method can be used for any CFG. If we slightly restrict the grammars, we can combine each match with the expand step just before, that introduced the terminal. This gives a very direct translation between grammar and its leftmost derivation, and a single state PDA and its computation.

On this normal form each production is of the form $A \to a\alpha$, where $a \in \Sigma \cup \{\Lambda\}$ can be the only terminal at the right. That means that any terminal pushed on the stack will be on top, and immediately will be matched.

Single state & empty stack

cfg G $\iff$ 1-pda M

$A \to \alpha$ $\quad$ $\delta(-, \Lambda, A) \ni (-, \alpha)$ expand

$\delta(-, a, a) = \{(-, \Lambda)\}$ match

normal form $\quad \alpha \in (\Sigma \cup \{\Lambda\}) \cdot V^*$

$A \to a\alpha$ $\quad$ $\delta(-, a, A) \ni (-, \alpha)$ combined

SimplePal: $S \to aSA \mid bSB \mid c$ $\quad$ $A \to a$ $\quad$ $B \to b$

leftmost derivation $\iff$ computation

	S		$(-, abbcbba, S)$
$\Rightarrow$	$a\,SA$	$\vdash$	$(-, bbcbba, SA)$
$\Rightarrow$	$ab\,SBA$	$\vdash$	$(-, bcbba, SBA)$
$\Rightarrow$	$abb\,SBBA$	$\vdash$	$(-, cbba, SBBA)$
$\Rightarrow$	$abbc\,BBA$	$\vdash$	$(-, bba, BBA)$
$\Rightarrow$	$abbcb\,BA$	$\vdash$	$(-, ba, BA)$
$\Rightarrow$	$abbcbb\,A$	$\vdash$	$(-, a, A)$
$\Rightarrow$	$abbcbba$	$\vdash$	$(-, \Lambda, \Lambda)$

In this case: deterministic PDA

Exercise 5.21.

Prove the converse of Theorem 5.28:
If there is a PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ accepting L by empty stack (that is, $x \in L$ if and only if $(q_0, x, Z_0) \vdash_M^* (q, \Lambda, \Lambda)$ for some state q), then there is a PDA M_1 accepting L by final state (i.e., the ordinary way).

Theorem

If $L = L_e(M)$ is the empty stack language of PDA M, then there exists a CFG G such that $L = L(G)$.

[M] Th 5.29

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

Theorem

If $L = L_e(M)$ is the empty stack language of PDA M, then there exists a CFG G such that $L = L(G)$.

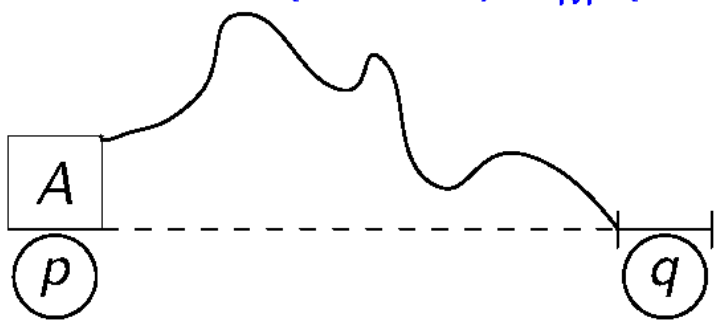
[M] Th 5.29

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

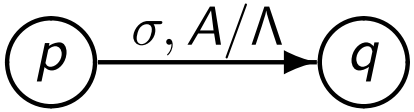
triplet construction

nonterminals $[p, A, q]$ $\quad p, q \in Q,\ A \in \Gamma$

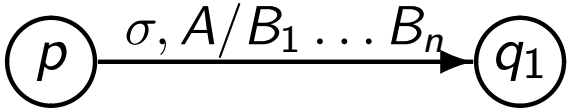
$[p, A, q] \Rightarrow_G^* w$ iff $(p, w, A) \vdash_M^* (q, \Lambda, \Lambda)$

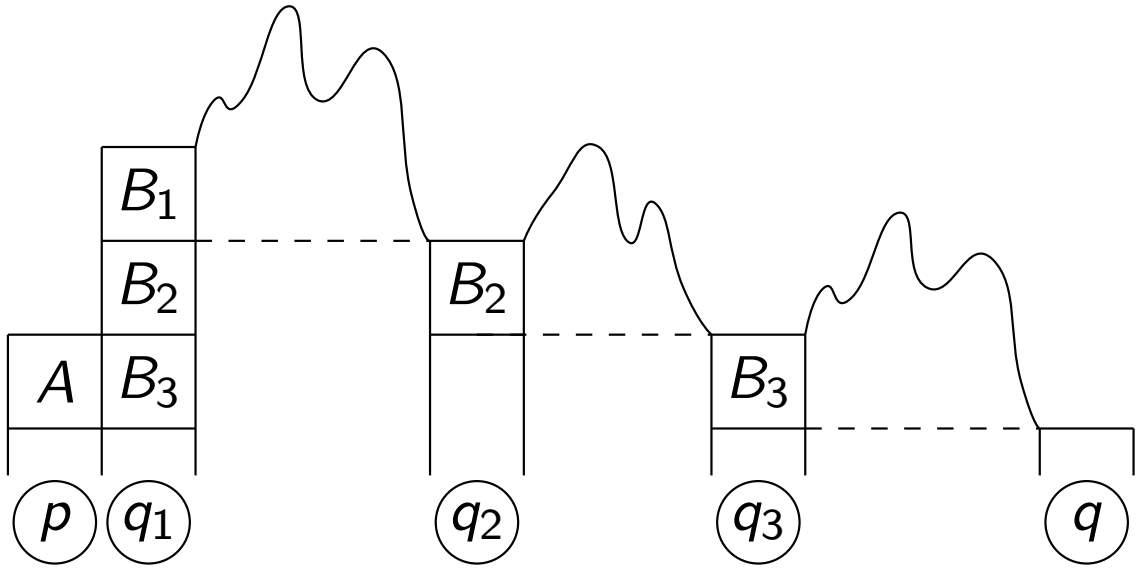


– productions



$[p, A, q] \to \sigma$ for $(q, \Lambda) \in \delta(p, \sigma, A)$





$[p, A, q] \to \sigma\ [q_1, B_1, q_2][q_2, B_2, q_3] \cdots [q_n, B_n, q]$

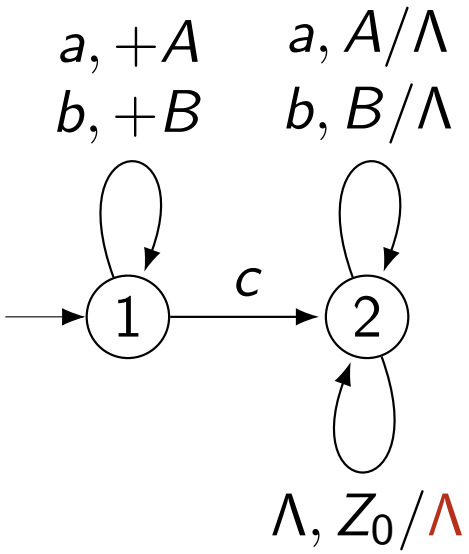
for $(q_1, B_1 \cdots B_n) \in \delta(p, \sigma, A)$, and $q, q_2, \ldots, q_n \in Q$

$S \to [q_0, Z_0, q]$ for all $q \in Q$

N.B.: σ may also be Λ

Construction from PDA to CFG, and the intuition behind it, must be known for the exam.
The details of the proof that $L(G) = L_e(M)$ do not have to be known for the exam.

Example

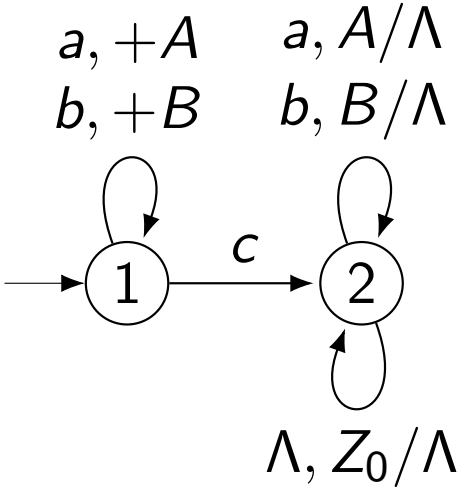


check stack

Example

$L_e(M) = SimplePal = \{\ wcw^r \mid w \in \{a, b\}^*\ \}$

12 transitions $\Rightarrow$ 33 $(+2)$ productions (!)



$$X \in \{A, B, Z_0\}$$

$$S \to [1, Z_0, 1] \mid [1, Z_0, 2]$$

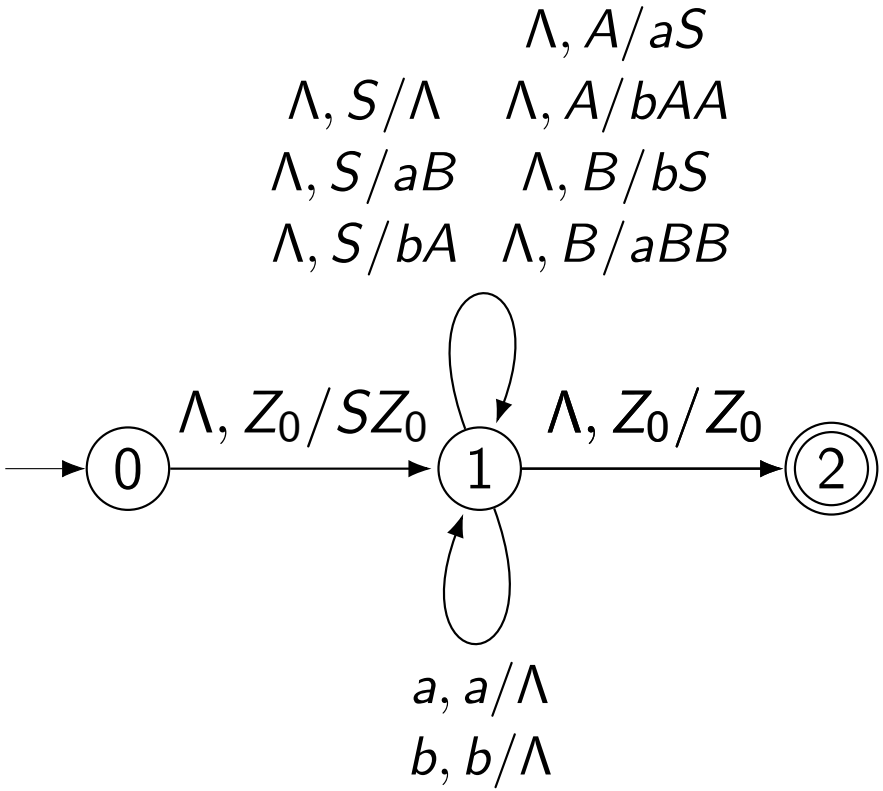
$\delta(1, a, X) = \{(1, AX)\}$	$[1, X, 1] \to a\,[1, A, 1][1, X, 1]$
	$[1, X, 1] \to a\,[1, A, 2][2, X, 1]$
	$[1, X, 2] \to a\,[1, A, 1][1, X, 2]$
	$[1, X, 2] \to a\,[1, A, 2][2, X, 2]$
$\delta(1, b, X) = \{(1, BX)\}$	...
$\delta(1, c, X) = \{(2, X)\}$	$[1, X, 1] \to c\,[2, X, 1]$
	$[1, X, 2] \to c\,[2, X, 2]$
$\delta(2, a, A) = \{(2, \Lambda)\}$	$[2, A, 2] \to a$
$\delta(2, b, B) = \{(2, \Lambda)\}$	$[2, B, 2] \to b$
$\delta(2, \Lambda, Z_0) = \{(2, \Lambda)\}$	$[2, Z_0, 2] \to \Lambda$

not 'live'

Example

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

$$S \to \Lambda \mid aB \mid bA$$
$$A \to aS \mid bAA$$
$$B \to bS \mid aBB$$



5.5. Parsing: make PDA (more) deterministic by looking ahead one symbol in input.
See Compiler Construction

Section 6

Context-Free and Non-Context-Free Languages

3 Context-Free and Non-Context-Free Languages
- Pumping Lemma
- Decision problems

Pumping lemma for regular languages

From Automaten:
Regular language is language accepted by an FA.

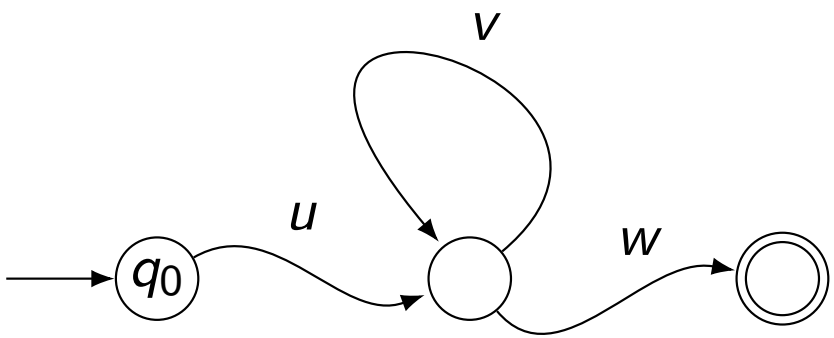
Theorem

Suppose L is a language over the alphabet Σ. If L is accepted by a finite automaton M, and if n is the number of states of M, then

$\forall$ *for every* $x \in L$
  satisfying $|x| \geq n$
$\exists$ *there are three string* u, v, *and* w,
  such that $x = uvw$ *and the following three conditions are true:*
  (1) $|uv| \leq n$,
  (2) $|v| \geq 1$
$\forall$ *and (3) for all* $i \geq 0$, $uv^i w$ *belongs to* L

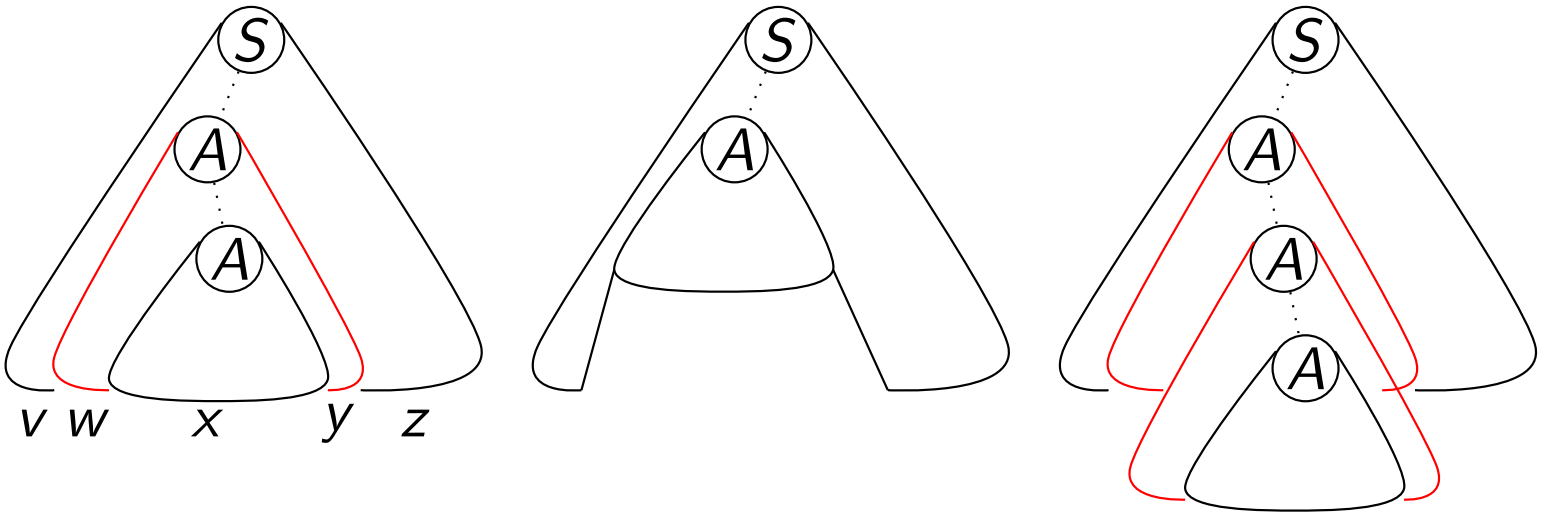
[M] Thm. 2.29

From Automaten:



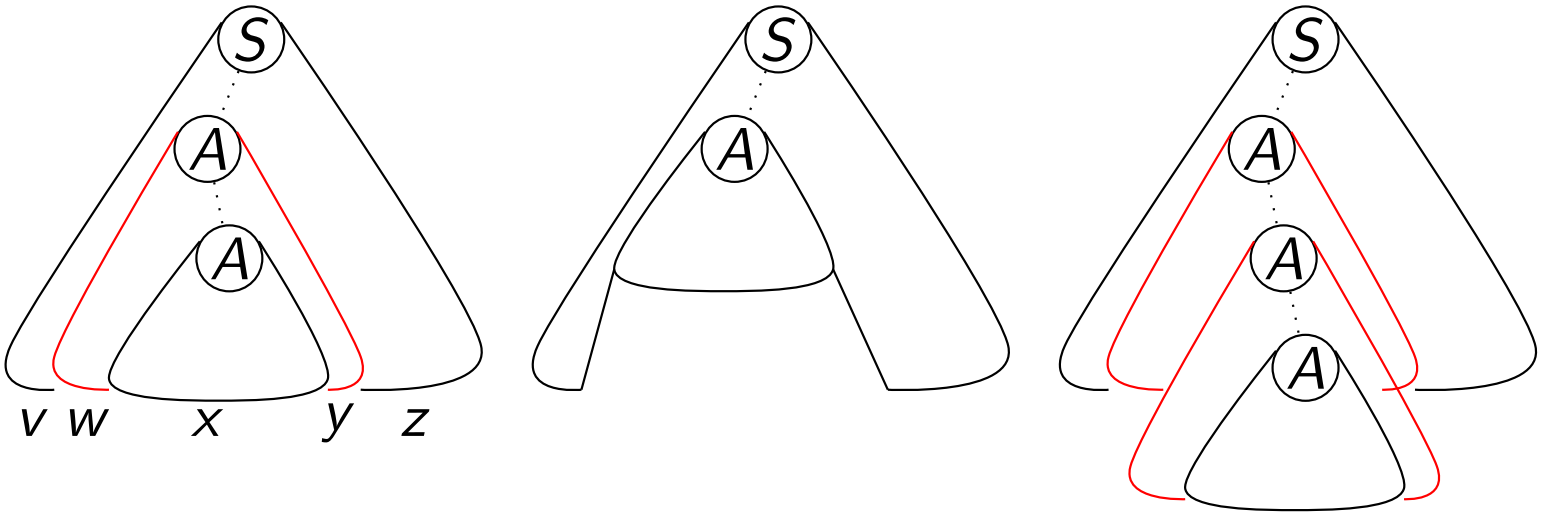
[M] Fig. 2.28

Pumping CF derivations



$S \Rightarrow^* vAz \Rightarrow^* vwAyz \Rightarrow^* vwxyz,\ v, w, x, y, z \in \Sigma^*$

$S \underset{(1)}{\Rightarrow}^* vAz,\ A \underset{(2)}{\Rightarrow}^* wAy,\ A \underset{(3)}{\Rightarrow}^* x$



$S \underset{(1)}{\Rightarrow}^* vAz \underset{(3)}{\Rightarrow}^* vxz$

$S \underset{(1)}{\Rightarrow}^* vAz \underset{(2)}{\Rightarrow}^* vwAyz \underset{(2)}{\Rightarrow}^* vwwAyyz \underset{(3)}{\Rightarrow}^* vwwxyyz$

Theorem (Pumping Lemma for context-free languages)

$\forall$ *for every context-free language* L

$\exists$ *there exists a constant* $n \geq 2$

such that

$\forall$ *for every* $u \in L$

with $|u| \geq n$

$\exists$ *there exists a decomposition* $u = vwxyz$

such that

(1) $|wy| \geq 1$

(2) $|wxy| \leq n$,

$\forall$ *(3) for all* $m \geq 0$, $vw^m xy^m z \in L$

[M] Thm. 6.1

Applying the Pumping Lemma

Example

$AnBnCn$ is not context-free.

[M] E 6.3

Example

AnBnCn is not context-free.

[M] E 6.3

$u = a^n b^n c^n$

Argument works also for $L = \{ x \in \{a, b, c\}^* \mid n_a(x) = n_b(x) = n_c(x) \}$