Lecture 7 – Pushdown automata

NTIN071 Automata and Grammars

Jakub Bulín (KTIML MFF UK)

Spring 2024

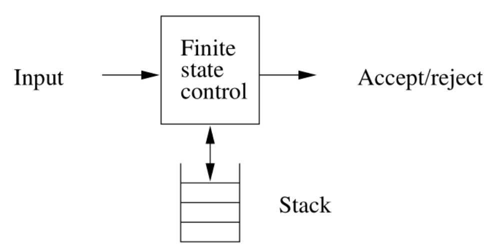
** Adapted from the Czech-lecture slides by Marta Vomlelová with gratitude. The translation, some modifications, and all errors are mine.*

Recap of Lecture 6

- Reducing a grammar: removing ϵ-productions, unit productions, useless symbols
- Chomsky Normal Form of a context-free grammar
- Pumping lemma for context-free languages, application: proving non-context-freeness
- Testing membership in a context-free language: the CYK algorithm

2.9 Pushdown automata

Pushdown automaton (PDA)



- an extension of ϵ–NFA, additional feature: a stack memory
- the stack has its own stack alphabet Γ (can contain Σ or not)
- at each step we pop the top stack symbol X, make a decision based on (q, a, X), push some word $\gamma \in \gamma^*$
- the stack can rememeber an infinite amount of information
- PDA define context-free languages, nondeterminism is important: deterministic PDA only recognize a proper subset of context-free languages (unlike DFA vs. NFA)

The definition

A pushdown automaton (PDA): $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where

- Q is finite, nonempty set of states
- Σ is a finite, nonempty input alphabet
- Γ is a finite, nonempty stack alphabet
- δ is the transition function,

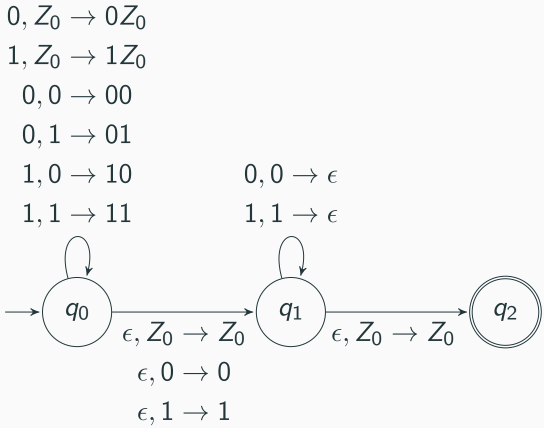
  $$\delta \colon Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_{FIN}(Q \times \Gamma^*)$$

  $\delta(q, a, X) \ni (p, \gamma)$ where p is the new state and γ a finite string of stack symbols that replace X on top of the stack
- $q_0 \in Q$ is the initial state
- $Z_0 \in \Gamma$ is the initial stack symbol (bottom of the stack); the only symbol on the stack at the beginning
- F is a set of accepting (final) states; may be undefined if our PDA accepts by empty stack

One transition of a PDA

- read one input letter ($a \in \Sigma$) or do an ϵ-transition ($a = \epsilon$)
- pop X from the top of the stack
- based on a, X, and the current state q nondeterministically choose one of finitely many options $(p, \gamma) \in \delta(q, a, X)$
- switch to the new state p
- push the finite string γ to the stack (the first symbol of Γ is now on top)
- pop: $\gamma = \epsilon$, read only: $\gamma = X$, push: $\gamma = \gamma' X$

Example: $L_{wwr} = \{ww^R \mid w \in \{0,1\}^*\}$



q_0 read input letters pushing them onto the stack; guess the middle (nondeterministically), jump to q_1

q_1 compare input with stack, consuming both; if empty stack (we see the bottom), accept by jumping to q_2; no input can remain

Example cont'd: full description of the PDA

$$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$$

$\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$ $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$	push input onto stack, leave the bottom
$\delta(q_0, 0, 0) = \{(q_0, 00)\}$ $\delta(q_0, 0, 1) = \{(q_0, 01)\}$ $\delta(q_0, 1, 0) = \{(q_0, 10)\}$ $\delta(q_0, 1, 1) = \{(q_0, 11)\}$	stay in q_0, push input onto stack
$\delta(q_0, \epsilon, Z_0) = \{(q_1, Z_0)\}$ $\delta(q_0, \epsilon, 0) = \{(q_1, 0)\}$ $\delta(q_0, \epsilon, 1) = \{(q_1, 1)\}$	jump to q_1 without changing stack
$\delta(q_1, 0, 0) = \{(q_1, \epsilon)\}$ $\delta(q_1, 1, 1) = \{(q_1, \epsilon)\}$	pop stack and match with input
$\delta(q_1, \epsilon, Z_0) = \{(q_2, Z_0)\}$	we have ww^R, go to accepting state

Notation

a, b, c	symbols of the input alphabet
q, p, r	states
u, w, x, y, z	words over input alphabet
X, Y, A, B	stack symbols
Z_0	bottom of the stack symbol
α, β, γ	words over stack alphabet

Transition diagram:

- nodes are states, initial and final denoted as usual
- a transition $\delta(q, a, X) \ni (p, \alpha)$: arc from p to q labelled $a, X \to \alpha$

The languages of a PDA

Configurations and moves (computation graph)

A configuration of a PDA is a triple (q, w, γ), where

q is the current state

w is the remaining input and

γ is the stack contents (the top is on the left)

We define moves between configurations ($\vdash_P$ or $\vdash$) thus: for any transition $\delta(q, a, X) \ni (p, \alpha)$ and all $w \in \Sigma^*$ and $\beta \in \Gamma^*$ we have

$$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$$

We use the symbol $\vdash_P^*$ or $\vdash^*$ to represent zero or more moves, i.e.

- $I \vdash^* I$ for any configuration I
- $I \vdash^* J$ if there exists K such that $I \vdash K$ and $K \vdash^* J$

Initial and accepting configurations, the languages of a PDA

The initial configuration of $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ for input word $w \in \Sigma^*$ is (q_0, w, Z_0). Which configurations are accepting?

Two options:

1. Acceptance by final state: (f, ϵ, γ) for some final state $f \in F$ and arbitrary stack contents $\gamma \in \Gamma^*$

$L(P) = \{w \mid (q_0, w, Z_0) \vdash_P^* (f, \epsilon, \gamma)$ for some $f \in F$ and $\gamma \in \Gamma^*\}$

2. Acceptance by empty stack: (q, ϵ, ϵ) for an arbitrary $q \in Q$

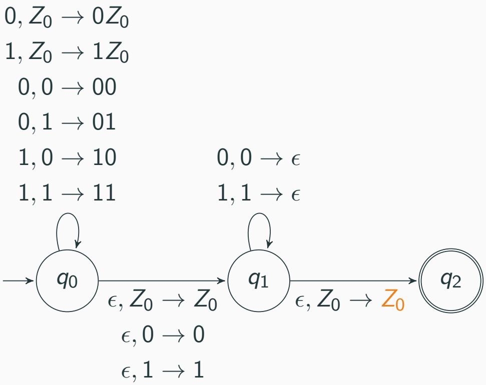
$N(P) = \{w \mid (q_0, w, Z_0) \vdash_P^* (q, \epsilon, \epsilon)$ for any $q \in Q\}$

In this case we can write only $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

Configurations for the input $w = 1111$

$$
\begin{tikzcd}
 & & & & \boxed{(q_0, 1111, Z_0)} \arrow[dl] \arrow[dr] & \\
 & & & (q_0, 111, 1Z_0) \arrow[dl] \arrow[dr] & & (q_1, 1111, Z_0) \arrow[d] \\
 & & (q_0, 11, 11Z_0) \arrow[dl] \arrow[dr] & & (q_1, 111, 1Z_0) \arrow[d] & (q_2, 1111, Z_0) \\
 & (q_0, 1, 111Z_0) \arrow[dl] \arrow[dr] & & (q_1, 11, 11Z_0) \arrow[d] & (q_1, 11, Z_0) \arrow[d] & \\
(q_0, \epsilon, 1111Z_0) \arrow[d] & & (q_1, 1, 111Z_0) \arrow[d] & (q_1, 1, 1Z_0) \arrow[d] & (q_2, 11, Z_0) & \\
(q_1, \epsilon, 1111Z_0) & & (q_1, \epsilon, 11Z_0) & (q_1, \epsilon, Z_0) \arrow[d] & & \\
 & & & \boxed{(q_2, \epsilon, Z_0)} & &
\end{tikzcd}
$$

Our example

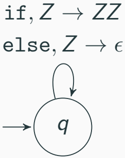


- acceptance by final state: $L(P) = L_{wwr}$
- to accept by empty stack: modify $\delta(q_1, \epsilon, Z_0) = \{(q_2, Z_0)\}$ to $\delta(q_1, \epsilon, Z_0) = \{(q_2, \epsilon)\}$ (erase bottom of the stack symbol), then also $N(P') = L_{wwr}$

Another example: if-else

Stop (accept) at first error, e.g. more `else`'s than `if`'s

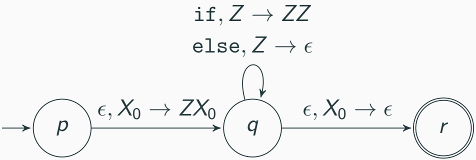
By empty stack: $P_N = (\{q\}, \{\texttt{if}, \texttt{else}\}, \{Z\}, \delta_N, q, Z)$



$$\delta_N(q, \texttt{if}, Z) = \{(q, ZZ)\} \ \text{(push)}$$

$$\delta_N(q, \texttt{else}, Z) = \{(q, \epsilon)\} \ \text{(pop)}$$

By final state: $P_F = (\{p, q, r\}, \{\texttt{if}, \texttt{else}\}, \{Z, X_0\}, \delta_F, p, X_0, \{r\})$



$$\delta_F(p, \epsilon, X_0) = \{(q, ZX_0)\} \ \text{(start)}$$

$$\delta_F(q, \texttt{if}, Z) = \{(q, ZZ)\} \ \text{(push)}$$

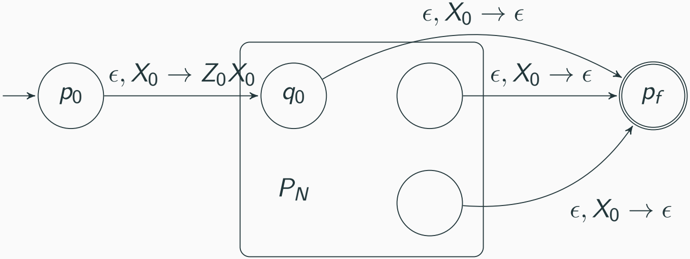
$$\delta_F(q, \texttt{else}, Z) = \{(q, \epsilon)\} \ \text{(pop)}$$

$$\delta_F(q, \epsilon, X_0) = \{(r, \epsilon)\} \ \text{(accept)}$$

From empty stack to final state

Lemma

If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, then there is a PDA P_F such that $L = L(P_F)$.



Idea: Make Z_0 a fake bottom (insert a new bottom X_0 below), so that we can tell when P_N's stack was empty. Add ϵ-transitions upon seeing X_0 from all states to a new, accepting state.

The proof

$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_F\})$, where δ_F is

- $\delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$ (start).
- $\forall(q \in Q, a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma)$, $\delta_F(q, a, Y) = \delta_N(q, a, Y)$.
- In addition, $\delta_F(q, \epsilon, X_0) \ni (p_f, \epsilon)$ for every $q \in Q$.

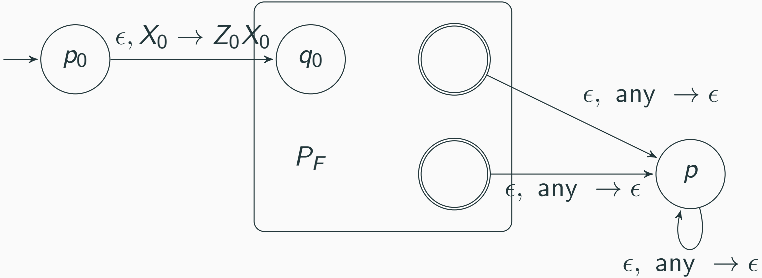
We must show that $w \in L(P_N)$ iff $w \in L(P_F)$.

- (If) P_F accepts as follows:
  $(p_0, w, X_0) \vdash_{P_F} (q_0, w, Z_0X_0) \vdash^*_{P_F=N_F} (q, \epsilon, X_0) \vdash_{P_F} (p_f, \epsilon, \epsilon)$.
- (Only if) No other way to go to p_F than the above. $\square$

From final state to empty stack

Lemma

If $L = L(P_F)$ for some PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$, then there exists a PDA P_N such that $L = N(P_N)$.



Idea: Make Z_0 a fake bottom (insert a new bottom below), because P_F could accidentally empty stack in a nonfinal state. Add ϵ-transitions (upon any stack symbol) from final states to a new state, there empty the stack without reading any input symbols.

The proof

Let $P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$, where

- $\delta_N(p_0, \epsilon, X_0) = \{(q, Z_0X_0)\}$ (start)
- $\forall(q \in Q, a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma)\ \delta_N(q, a, Y) = \delta_F(q, a, Y)$ (simulate)
- $\forall(q \in F, Y \in \Gamma \cup \{X_0\}),\ \delta_N(q, \epsilon, Y) \ni (p, \epsilon)$ (i.e. accept if P_F accepts)
- $\forall(Y \in \Gamma \cup \{X_0\}), \delta_N(p, \epsilon, Y) = \{(p, \epsilon)\}$ clean the stack.

The proof $w \in N(P_N)$ iff $w \in L(P_F)$ is similar as before. $\square$

Unseen data cannot affect computation

Lemma

If $(q, x, \alpha) \vdash_P (p, y, \beta)$*, then for any* $w \in \Sigma^*$ *and* $\gamma \in \Gamma^*$ *we also have* $(q, xw, \alpha\gamma) \vdash_P^* (p, yw, \beta\gamma)$*. (In particular,* $\gamma = \epsilon$ *or* $w = \epsilon$*.)*

Proof: Induction on the number length of the sequence of configurations that take $(q, xw, \alpha\gamma)$ to $(p, yw, \beta\gamma)$. Each of the moves $(q, x, \alpha) \vdash_P^* (p, y, \beta)$ is justified without using w and/or γ in any way. The moves are still valid with w, γ on the input/stack. □

Lemma

If $(q, xw, \alpha) \vdash_P^* (p, yw, \beta)$*, then also* $(q, x, \alpha) \vdash_P^* (p, y, \beta)$*.*

NB: Not true for stack, the computation may require γ on the stack and then push it back. (E.g. $L = \{0^i 1^i 0^j 1^j\}$, configuration $(p, 0^{i-j} 1^i 0^j 1^j, 0^j Z_0) \vdash^* (q, 1^j, 0^j Z_0)$, inbetween clear the stack.)

2.10 Equivalence of PDA and context-free grammars

Equivalence of PDA and CFG

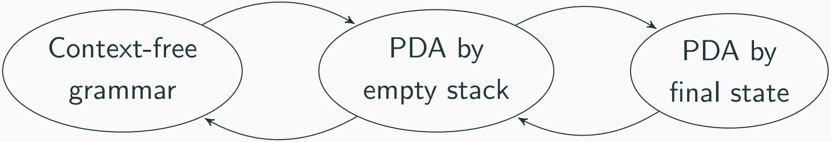
Theorem

The following statements about $L \subset \Sigma^*$ are equivalent:

(i) *There exists a context-free grammar such that $L(G) = L$.*

(ii) *There exists a PDA such that $L(P) = L$.*

(iii) *There exists a PDA such that $N(P) = L$.*



We have already shown $(ii) \Leftrightarrow (iii)$. To prove equivalence with a context-free grammar, we use acceptance by empty stack.

Context-free grammar to pushdown automaton

CFG to PDA

The construction

Given $G = (V, T, \mathcal{P}, S)$, construct $P = (\{q\}, T, V \cup T, \delta, q, S)$:

(1) for each $A \in V$, $\delta(q, \epsilon, A) = \{(q, \beta) \mid A \to \beta \in \mathcal{P}\}$

[apply rule]

(2) for each $a \in T$, $\delta(q, a, a) = \{(q, \epsilon)\}$

[match terminal]

How it works:

- a leftmost derivation is simulated by the PDA
- current sentential form = part of input read + stack contents
- see a variable: apply rule, a terminal: read & pop from stack

An example

Example

$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$,
$E \to I \mid E * E \mid E + E \mid (E)$

$\Sigma = \{a, b, 0, 1, (,), +, *\}$, $\Gamma = \Sigma \cup \{I, E\}$, δ is defined as follows:

- $\delta(q, \epsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\}$
- $\delta(q, \epsilon, E) = \{(q, I), (q, E * E), (q, E + E), (q, (E))\}$
- $\delta(q, s, s) = \{(q, \epsilon)\}$ for all $s \in \Sigma$ (e.g. $\delta(q, +, +) = \{(q, \epsilon)\}$)
- $\delta(q, x)$ is empty otherwise

Leftmost derivation: $E \Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow a * I \Rightarrow a * b$

The sequence of configurations:

$(q, a * b, E) \vdash (q, a * b, E * E) \vdash (q, a * b, I * E) \vdash (q, a * b, a * E)$
$\vdash (q, *b, *E) \vdash (q, b, E) \vdash (q, b, I) \vdash (q, b, b) \vdash (q, \epsilon, \epsilon)$

Proof that $N(P) = L(G)$ **(i)** $w \in L(G) \Rightarrow w \in N(P)$

Start with a leftmost derivation $S = \gamma_1 \Rightarrow_{lm} \ldots \Rightarrow_{lm} \gamma_n = w$.

Prove by induction on i that $(q, w, S) \vdash_P^* (q, v_i, \alpha_i)$, where $\gamma_i = u_i\alpha_i$ is the i-th sentential form and $u_i v_i = w$.

If γ_i contains only terminals, set $\gamma_i = w = u_i, v_i = \epsilon = \alpha_i$. Otherwise, write $\gamma_i = u_i A \alpha_i$, where $u_i \in T^*$ and $A \in V$ is the leftmost variable.

By induction we have $(q, w, S) \vdash_P^* (q, v_i, A\alpha_i)$, $w = u_i v_i$.

For the step $\gamma_i \Rightarrow_{lm} \gamma_{i+1}$ we used some rule $A \to \beta \in P$. The PDA replaces A on the stack with β, moves to configuration $(q, v_i, \beta\alpha_i)$.

We pop all terminals $v \in \Sigma^*$ from the beginning of $\beta\alpha$ (matching them with the input): $v_i = v v_{i+1}$ and $\beta\alpha = v\alpha_{i+1}$

We got to $(q, v_{i+1}, \alpha_{i+1})$, corresponds to the sentential form γ_{i+1}.

Proof that $N(P) = L(G)$ (ii) $w \in N(P) \Rightarrow w \in L(G)$

Prove that if $(q, u, X) \vdash_P^* (q, \epsilon, \epsilon)$, then $X \Rightarrow_G^* u$. By induction on the number of moves. **Basis** $n = 1$ move:

- $X = a \in \Sigma$: $\delta(q, a, a) \ni (q, \epsilon)$, $u = a$, 0-step derivation
- $X = A \in \Gamma$: $\delta(q, \epsilon, A) \ni (q, \epsilon)$ coming from $A \to \epsilon \in \mathcal{P}$, $u = \epsilon$

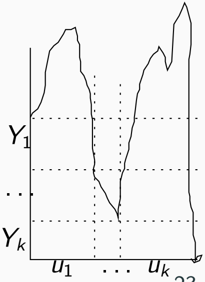
Induction step $n > 1$ moves: if the first move is [match terminal], don't extend the derivation, if it is [apply rule]: A on top of stack was replaced by $\beta = Y_1 Y_2 \dots Y_k$, for a rule $A \to \beta \in \mathcal{P}$.

Split $u = u_1 \dots u_k$ s.t. while popping Y_i we read u_i, i.e. $(q, u_i u_{i+1} \dots u_k, Y_i) \vdash^* (q, u_{i+1} \dots u_k, \epsilon)$

Thus also $(q, u_i, Y_i) \vdash^* (q, \epsilon, \epsilon)$, by induction assumption we get $Y_i \Rightarrow^* u_i$. Together:

$$A \Rightarrow Y_1 Y_2 \dots Y_k \Rightarrow^* u_1 Y_2 \dots Y_k \Rightarrow^* \dots \Rightarrow^* u_1 u_2 \dots u_k$$

$\square$



Pushdown automaton to context-free grammar

PDA to CFG

The construction

Given $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$, construct $G = (V, T, \mathcal{P}, S)$ where $V = \{S\} \cup \{[pXq] \mid p, q \in Q, X \in \Gamma\}$ and the productions are:

(i) for every state $p \in Q$ add $S \to [q_0 X p]$

(ii) for every transition $(p, Y_1 Y_2 \dots Y_k) \in \delta(q, a, X)$ (incl. $a = \epsilon$) and all k-tuples of states $p_1, \dots, p_{k-1}, p_k \in Q$ add

$$[qXp_k] \to a[pY_1p_1][p_1Y_2p_2]\dots[p_{k-1}Y_kp_k]$$

In particular, for $(p, \epsilon) \in \delta(q, a, X)$ (i.e., $k = 0$) add $[qXp] \to a$.

- key event: pop a symbol X, while changing from state q to r
- variables: $[qXr]$ for $q, r \in Q$ and $X \in \Gamma$, plus a new variable S

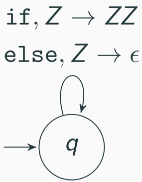
$$L([qXr]) = \{w \in \Sigma^* \mid (q, w, X) \vdash_P^* (r, \lambda, \lambda)\}$$

- S to choose (guess) in which state the stack is emptied

An example

Given $P = (\{q\}, \{\texttt{if}, \texttt{else}\}, \{Z\}, \delta, q, Z)$

$$\delta(q, \texttt{if}, Z) = \{(q, ZZ)\}$$
$$\delta(q, \texttt{else}, Z) = \{(q, \epsilon)\}$$



Construct $G = (V, \{\texttt{if}, \texttt{else}\}, \mathcal{P}, S)$

- variables: $V = \{S, [qZq]\}$
- production rules:
    - $S \to [qZq]$
    - $[qZq] \to \texttt{else}$
    - $[qZq] \to \texttt{if}[qZq][qZq]$

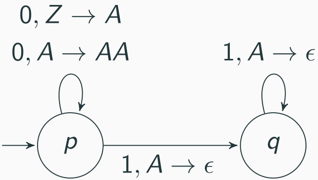
In this example, S and $[qZq]$ generate the same words, so we can simplify: $G = (\{S\}, \{\texttt{if}, \texttt{else}\}, \{S \to \texttt{if}SS \mid \texttt{else}\}, S)$

Another example: $\{0^n 1^n \mid n > 0\}$

δ	Productions	
	$S \to [pZp] \mid [pZq]$	(1)
$\delta(p, 0, Z) \ni (p, A)$	$[pZp] \to 0[pAp]$	(2)
	$[pZq] \to 0[pAq]$	(3)
$\delta(p, 0, A) \ni (p, AA)$	$[pAp] \to 0[pAp][pAp]$	(4)
	$[pAp] \to 0[pAq][qAp]$	(5)
	$[pAq] \to 0[pAp][pAq]$	(6)
	$[pAq] \to 0[pAq][qAq]$	(7)
$\delta(p, 1, A) \ni (q, \epsilon)$	$[pAq] \to 1$	(8)
$\delta(q, 1, A) \ni (q, \epsilon)$	$[qAq] \to 1$	(9)

Derivation of 0011:

$$S \Rightarrow^{(1)} [pZq] \Rightarrow^{(3)} 0[pAq]$$
$$\Rightarrow^{(7)} 00[pAq][qAq]$$
$$\Rightarrow^{(8)} 001[qAq] \Rightarrow^{(9)} 0011$$

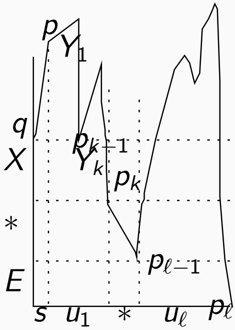


Sketch of proof that $L(G) = N(P)$

It suffices to show that:

$$[qXp] \Rightarrow^* w \ \text{ iff } \ (q, w, X) \vdash^* (p, \epsilon, \epsilon)$$

In both directions, the proof is done by induction (number of moves/steps).



$\square$

Summary of Lecture 7

- Pushdown automaton: extend an ϵ-NFA with a stack memory (potentially infinite), pop the top symbol, decide based on (q, a, X), can push a finite string of stack symbols
- Acceptance by final state $L(P)$ and by empty stack $N(P)$, conversion between the two options
- Pushdown automata accept exactly context-free languages (constructions: CFG to PDA and PDA to CFG)