# Lecture 8 – Equivalence of PDA and CFG, Deterministic PDA

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 7

- Testing membership in a context-free language: the Cocke-Younger-Kasami algorithm
- Testing emptiness and finiteness of a context-free language
- Pushdown automaton: extend an $\epsilon$-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

# 2.10 Equivalence of PDA and context-free grammars

# Equivalence of PDA and CFG

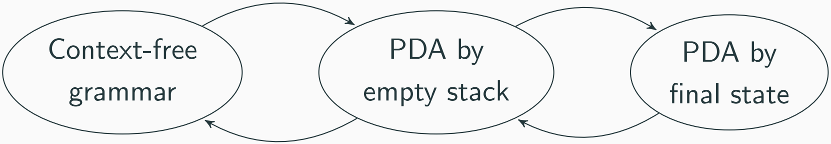
**Theorem**

*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**

$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1,$

$E \to I \mid E * E \mid E + E \mid (E)$

$\Sigma = \{a, b, 0, 1, (, ), +, *\}$, $\Gamma = \Sigma \cup \{I, E\}$, $\delta$ 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:

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

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 $\gamma_i$ 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 $\beta$, 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 $\gamma_{i+1}$.

## 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 $A \to \epsilon \in \mathcal{P}$, $u = \epsilon$

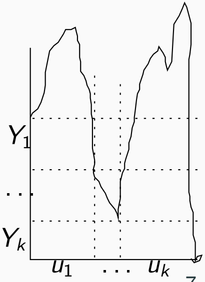
**Induction step** $n > 1$ moves: if the first move is [match terminal], don't extend the derivation, if it is [apply rule]: $A$ on top of stack was replaced by $\beta = Y_1 Y_2 \ldots Y_k$, for a rule $A \to \beta \in \mathcal{P}$.

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

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

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

$\square$

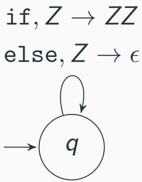


# Pushdown automaton to context-free grammar

# 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: $G = (\{S\}, \{\texttt{if}, \texttt{else}\}, \{S \to \texttt{if}SS \mid \texttt{else}\}, S)$

# PDA to CFG: the construction

- 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 $S$

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

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

**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 $S \to [q_0Xp]$

(ii) for every transition $(p, Y_1Y_2 \ldots Y_k) \in \delta(q, a, X)$ (incl. $a = \epsilon$) and all $k$-tuples of states $p_1, \ldots, p_{k-1}, p_k \in Q$ add

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

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

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

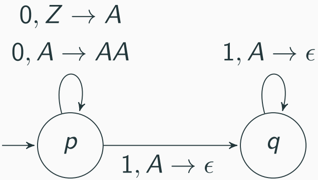
| $\delta$ | Productions | |
|---|---|---|
| | $S \to [pZp] \mid [pZq]$ | (1) |
| $\delta(p, 0, Z) \ni (p, A)$ | $[pZp] \to 0[pAp]$ | (2) |
| | $[pZq] \to 0[pAq]$ | (3) |
| $\delta(p, 0, A) \ni (p, AA)$ | $[pAp] \to 0[pAp][pAp]$ | (4) |
| | $[pAp] \to 0[pAq][qAp]$ | (5) |
| | $[pAq] \to 0[pAp][pAq]$ | (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:

$$S \Rightarrow^{(1)} [pZq] \Rightarrow^{(3)} 0[pAq]$$

$$\Rightarrow^{(7)} 00[pAq][qAq]$$

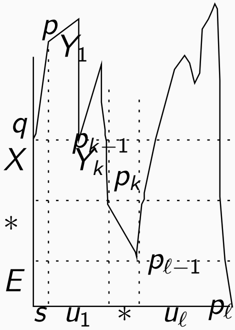
$$\Rightarrow^{(8)} 001[qAq] \Rightarrow^{(9)} 0011$$



It suffices to show that:

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

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




$\square$

# 2.11 Deterministic pushdown automata

# The definition

**Definition (Deterministic PDA)**

A pushdown automaton $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is deterministic (a DPDA) iff both of the following hold:

(i) The set of possible transitions $\delta(q, a, X)$ is at most one-element for all $q \in Q$, $a \in \Sigma \cup \{\epsilon\}$, and $X \in \Gamma$.

(ii) If $\delta(q, a, X) \neq \emptyset$ for some $a \in \Sigma$, then $\delta(q, \epsilon, X) = \emptyset$.

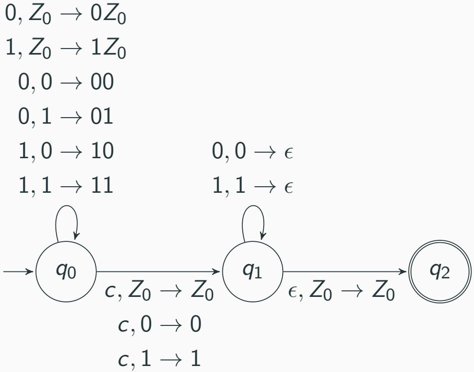
A language $L$ is a deterministic context-free language, if $L = L(P)$ for some DPDA $P$.

(Reasonable programming languages are deterministic.)

# Example: even palindromes with a center mark

$L_{wwr} = \{ww^R \mid w \in \{0,1\}^*\}$ is context free, but not recognizable by a DPDA [proof coming soon]; (ii) forbids a choice between an $\epsilon$-transition and reading the next input symbol.

But if we put in a 'center mark' $c$, $L_{wcwr} = \{wcw^R \mid w \in \{0,1\}^*\}$ is recognized by the following DPDA:



# Languages recognized by deterministic PDA

$$RL \subsetneq L_{DPDA} \subsetneq L_{PDA} = CFL = N_{PDA} \supsetneq N_{DPDA}$$

**Proposition**

*For every regular language $L$, there is a DPDA $P$ with $L = L(P)$.*

DPDA can simulate a DFA, ignore the stack (leave $Z_0$ on top). $\square$

**Example**

$L_{wcwr}$ is recognized by a DPDA by final state, but not regular.

DPDA on the previous slide, nonregularity by PL for $z = 0^n c 0^n$. $\square$

**Example**

$L = \{a^i b^i \mid i \in \mathbb{N}\} \cup \{a^i b^{2i} \mid i \in \mathbb{N}\}$ is context-free, but not recognizable by any DPDA by final state.

Context-freeness is easy. Not in $L_{DPDA}$: next slide.

# $L_{abb} = \{a^i b^i | i \in \mathbb{N}\} \cup \{a^i b^{2i} | i \in \mathbb{N}\} \notin L_{DPDA}$

Assume for contradiction: recognized by a DPDA $M$ by final state.
Create two copies, $M_1$ and $M_2$, call corresponding nodes 'siblings'.
Construct a (nondeterministic) PDA $M'$:

- initial state: the initial state of $M_1$
- final states: the final states of $M_2$
- reroute transitions going out of final states of $M_1$ to the siblings in $M_2$, relabel $b$ to $c$ (e.g. $\delta(f, b, X) = \{(q, X)\}$ becomes $\delta(f, c, X) = \{(q', X)\}$ where $q'$ is the sibling of $q$)
- in the automaton $M_2$, relabel $b$-transitions to $c$-transitions

The resulting PDA $M'$ recognizes $\{a^i b^i c^i \mid i \in \mathbb{N}\}$ by final state: determinism of $M$ means a unique path when reading $a^i b^{2i}$, thus after the initial $a^i b^i$, $M$ is in an accepting state. Then $M'$ continues reading $c^i$, ends in a final state in $M_2$, and accepts.

But we know $\{a^i b^i c^i | i \in \mathbb{N}\}$ is not context-free, a contradiction. $\square$

# Prefix-free languages & acceptance by empty stack

**Definition**

A language $L \subseteq \Sigma^*$ is prefix-free if there are no words $u, v \in L$ such that $u$ is a proper prefix of $v$ (i.e., $u = vz$ for some $z \in \Sigma^+$).

For example, $L_{wcwr}$ is prefix-free while $L_{wwr}$ is not.

**Theorem**

*For any language $L$, $L = N(P)$ for some DPDA $P$ if and only if $L$ is prefix-free and $L = L(P')$ for some DPDA $P'$.*

**Proof:** $\Rightarrow$ The prefix $u$ is accepted by empty stack. There is no transition for empty stack, so no proper extension $v$ of $u$ can be in $N(P) = L$. Thus $L$ is prefix-free. The conversion from empty stack to final state acceptance does not add nondeterminism.

$\Leftarrow$ Since $L$ is prefix-free, we can delete transitions out of final states. Then the conversion to empty stack acceptance does not add nondeterminism. $\square$

# Deterministic PDA have unambiguous grammars

**Theorem**

*If $L$ is recognized by a DPDA (either by final state or empty stack), then $L$ has an unambiguous grammar.*

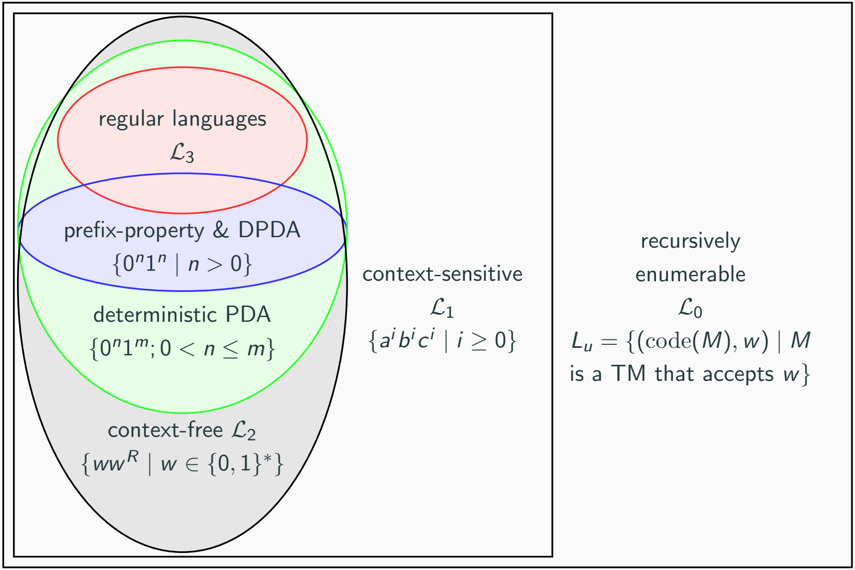
**NB:** The converse is not true, $L_{wwr}$ has an unambiguous grammar $S \to 0S0 \mid 1S1 \mid \epsilon$ but is not a DPDA language.

**Proof:** $L = N(P)$: PDA (by empty stack) to CFG construction, when applied to a DPDA, yields an unambiguous grammar.

$L = L(P)$: First, convert $L$ to a prefix-free language $L'$ by adding a new symbol \$ at the end of each word. Now, construct a DPDA $P'$ such that $L' = N(P')$. Convert it to an unambiguous grammar $G'$ generating the language $N(P') = L'$.

Finally, modify $G'$ to get a grammar $G$ for $L$: Change the terminal \$ to a nonterminal, add the rule $\$ \to \epsilon$. Note that $G$ is unambiguous since $G'$ was, and we did not add ambiguity. $\square$

# The landscape of languages




$L_d = \{w \mid$ the TM with code $w$ rejects input $w\}$

# Converting between representations of context-free languages

**Conversions linear in input size:**

- CFG to a PDA
- PDA by final state to a PDA by empty stack
- PDA by empty stack to PDA by final state

**PDA to CFG:** $O(n^3)$ algorithm that takes a PDA $P$ whose representation has length $n$ and produces a CFG of length (sum of sizes of bodies) at most $O(n^3)$

**Conversion to ChNF:** given a $\epsilon$-production-free grammar $G$ of length $n$, we can find an equivalent ChNF grammar for G in time $O(n^2)$; the resulting grammar has length $O(n^2)$

# Undecidable problems about context-free languages (preview)

The following problems are not algorithmically decidable:

- Is a given context-free grammar ambiguous?
- Is a given context-free language inherently ambiguous?
- Is the intersection of two context-free languages empty?
- Is a given context-free language equal to $\Sigma^*$?

## Summary of Lecture 8

- Pushdown automata accept exactly context-free languages (constructions: CFG to PDA and PDA to CFG)
- A deterministic pushdown automaton (DPDA)
- DPDA recognize a proper subclass of context-free languages, accepts by empty stack iff prefix-free and accepts by final state (Deterministic PDA + acceptance by empty stack does not even cover regular languages!)
- Deterministic PDA have unambiguous grammars
- The landscape of languages
- Converting between representations of context-free languages
- Undecidable problems about context-free languages (preview)