

# TDDD14 / TDDD85 – Lecture 10

## Equivalence between CFG and PDA

August Ernstsson, 2026 (based on lecture notes by Jonas Wallgren)

# From previous lectures

# Definition 1: Context-free grammars

- A *grammar* is a quadruple  $G = \langle N, \Sigma, P, S \rangle$  where
  - $N$  = set of *nonterminals*
  - $\Sigma$  = set of *terminals* (the alphabet)
  - $P \subseteq N \times (N \cup \Sigma)^*$  = set of *production rules*
  - $S \in N$  = *start symbol*
- $P$  is a set of elements with
  - A left-hand side that is a nonterminal
  - A right-hand side that is a mix of terminals and nonterminals

# Derivations

- Performing a derivation
  - Begin with the start symbol S;
  - step by step, use the production rules in P;
  - finally, end up with a string of terminals.
- In general, we use small greek letters for sequences of nonterminals and terminals.
  - **Example:**  $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$
- Capital latin letters stand for nonterminals.

# Left and right (-most) derivations

- We will consider the grammar  $E \rightarrow E^*E \mid E+E \mid a \mid b \mid c$
- An example derivation
  - $\underline{E} \Rightarrow \underline{E^*}E \Rightarrow \underline{E+E^*}E \Rightarrow a+\underline{E^*}E \Rightarrow a+b^*\underline{E} \Rightarrow a+b^*c$
  - In each step, the *leftmost* nonterminal has been chosen.
  - Called a *leftmost derivation*, symbol:  $\Rightarrow_{lm}$
- Another derivation
  - $\underline{E} \Rightarrow \underline{E+E} \Rightarrow \underline{E+E^*}E \Rightarrow \underline{E+E^*}c \Rightarrow \underline{E}+b^*c \Rightarrow a+b^*c$
  - In each step, the *rightmost* nonterminal has been chosen.
  - Called a *rightmost derivation*, symbol:  $\Rightarrow_{rm}$

# Definition 1: Greibach normal form

- A grammar is in Greibach normal form (GNF)
  - **if all** productions have the form  $A \rightarrow aB_1B_2\dots B_n$
  - (Sequence of B's may be empty)
- Every CFG can be converted to a CFG in GNF.
- Greibach normal form will be important in lecture 10.

# Definition 1: Pushdown Automata

- A PDA is a septuple  $\langle Q, \Sigma, \Gamma, \delta, s, \perp, F \rangle$ 
  - $Q$  = set of states
  - $\Sigma$  = (input) alphabet
  - $\Gamma$  = stack alphabet
  - $s$  = start state  $\in Q$
  - $F$  = final states  $\subseteq Q$
  - $\perp$  = start stack symbol
  - $\delta$  = transition relation  $\subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$

# The transition relation $\delta$

- $\delta = \text{transition relation} \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$
- $\langle \langle p, a, A \rangle, \langle q, B_1 B_2 \dots B_n \rangle \rangle \in \delta$  means:
  - In state  $p$  with  $A$  on top of the stack:
    - Read  $a$ ,
    - go to state  $q$ ,
    - change the stack top  $A$  to  $B_1 B_2 \dots B_n$  (left is top of stack).

# The transition relation $\delta$

- $\delta = \text{transition relation} \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$
- $\langle \langle p, \varepsilon, A \rangle, \langle q, B_1 B_2 \dots B_n \rangle \rangle \in \delta$  means:
  - In state  $p$  with  $A$  on top of the stack, without reading anything:
    - Go to state  $q$ ,
    - change the stack top  $A$  to  $B_1 B_2 \dots B_n$  (the left end being the new stack top).

# Acceptance

- Two different modes of acceptance are used in the literature.
- A PDA can accept a string if
  - the *stack is empty*, or
  - if it reaches a *final state*.
- **Definition 4**
  - A PDA accepts the string  $x$  if  $\langle s, x, \perp \rangle \xrightarrow{*} \langle q, \varepsilon, \gamma \rangle$  when  $q \in F$ .
- **Definition 5**
  - A PDA accepts the string  $x$  if  $\langle s, x, \perp \rangle \xrightarrow{*} \langle q, \varepsilon, \varepsilon \rangle$  when  $q \in Q$

# Configurations

- What we handled in Example 2 are called *configurations*.
- **Definition 2.** A configuration is a triple  $\langle q, x, \gamma \rangle$ , where:
  - $q = \text{state} \in Q$
  - $x = \text{string} \in \Sigma^*$
  - $\gamma = \text{stack} \in \Gamma^*$
- **Definition 3.** The next-configuration relation  $\rightarrow$ 
  - If  $\langle \langle p, a, A \rangle, \langle q, \gamma \rangle \rangle \in \delta$  then  $\langle p, ay, A\beta \rangle \rightarrow \langle q, y, \gamma\beta \rangle$
  - If  $\langle \langle p, \varepsilon, A \rangle, \langle q, \gamma \rangle \rangle \in \delta$  then  $\langle p, y, A\beta \rangle \rightarrow \langle q, y, \gamma\beta \rangle$

# Notation

$\rightarrow$  BNF-notation for productions in CFGs:  $A \rightarrow B$  etc.

$\rightarrow$  PDA state transitions

$\rightarrow_M$  PDA state transitions, PDA  $M$

$_n\rightarrow$  PDA state transitions,  $n$  steps

$*\rightarrow$  PDA state transitions, any number of steps

$\Rightarrow$  derivation of a string in CFG

$*\Rightarrow$  derivation of a string in CFG,  $n$  steps

$\Rightarrow_{lm}$  leftmost derivation of a string in CFG

Let's start!

# Today's topic

- In this lecture, we will show that context free grammars and pushdown automata *have the same power*.
  - Equivalence between CFG and PDA.
- Comment on DPDAs if there is time.
  - We will return to PDAs later when we compare different classes of languages.

# Introduction

- In lecture 7, we introduced **CFGs** to express languages with strings that can't be described using regular expressions.
- In lecture 9, we introduced **PDAs** for the same purpose.
- Like we did for regular languages, we will now prove that the automata and the notation for this new class of languages (CFLs) **are equivalent**.
- This is done in two steps:
  1. The equivalence is proved as an implication in one direction (CFG to PDA),
  2. Then the implication in the other direction (PDA to CFG).

# CFG to PDA

- We prove that **given** a CFG, we can **construct** a PDA that accepts the language of the CFG.
- The starting point is a grammar  $G = \langle N, \Sigma, P, S \rangle$  on **Greibach normal form**,
  - e.g. every rule in  $P$  has the form  $A \rightarrow cB_1B_2\dots B_n$  (where  $c \in \Sigma$ ,  $B_i \in N$ ,  $n \geq 0$ ).
- Now, create a PDA  $M = \langle \{q\}, \Sigma, N, \delta, q, S, \emptyset \rangle$ , where
  - There is just **one** state:  $Q = \{q\}$
  - The **stack symbols are the nonterminals**:  $\Gamma = N$ 
    - with the grammar start symbol as the stack start symbol:  $\perp = S$
  - There are no final states, so **acceptance** will be **by empty stack**.

# CFG to PDA, the idea

- **Recall:** Greibach normal form  $A \rightarrow aB_1B_2\dots B_n$
- When there is a nonterminal on the stack top:
  1. Choose one of the production rules for that nonterminal,
  2. read the "a" first in its right-hand side,
  3. "queue" all the B:s on the stack.
- So,  $\delta$  will for each grammar rule contain  $\langle\langle q, a, A \rangle, \langle q, B_1B_2\dots B_n \rangle\rangle$

# Example 1

- A GNF grammar for arithmetic expressions:

$$G = \langle N, \Sigma, P, S \rangle = \langle \{E, T, X, Y\}, \{a, b, +, *\}, P, E \rangle$$

- P is the set of production rules:

$$E \rightarrow aX \mid bX \mid aY \mid bY \mid aYX \mid bYX \mid a \mid b$$

$$T \rightarrow aY \mid bY \mid a \mid b$$

$$X \rightarrow +E$$

$$Y \rightarrow *T$$

- A PDA from the grammar above:

- $M = \langle Q, \Sigma, \Gamma, \delta, s, \perp, F \rangle = \langle \{q\}, \{a, b, +, *\}, \{E, T, X, Y\}, \delta, q, E, \emptyset \rangle$

**5 of the 14 elements in  $\delta$ :**

$$\langle \langle q, a, E \rangle, \langle q, Y X \rangle \rangle,$$

$$\langle \langle q, *, Y \rangle, \langle q, T \rangle \rangle,$$

$$\langle \langle q, b, T \rangle, \langle q, \epsilon \rangle \rangle,$$

$$\langle \langle q, +, X \rangle, \langle q, E \rangle \rangle,$$

$$\langle \langle q, a, E \rangle, \langle q, \epsilon \rangle \rangle.$$

# Example 1, continued

$$E \rightarrow aX \mid bX \mid aY \mid bY \mid aYX \mid bYX \mid a \mid b$$

$$T \rightarrow aY \mid bY \mid a \mid b$$

$$X \rightarrow +E$$

$$Y \rightarrow *T$$

- Performing the (leftmost) derivation ...

- $E \Rightarrow a Y X \Rightarrow a * T X \Rightarrow a * b X \Rightarrow a * b + E \Rightarrow a * b + a$

- ... corresponds to the following next-configuration steps using the PDA:

$$\langle q, a * b + a, E \rangle \rightarrow$$

$$\langle q, * b + a, Y X \rangle \rightarrow$$

$$\langle q, b + a, T X \rangle \rightarrow$$

$$\langle q, + a, X \rangle \rightarrow$$

$$\langle q, a, E \rangle \rightarrow$$

$$\langle q, \varepsilon, \varepsilon \rangle$$

- and the string is accepted with empty stack!

Some items in  $\delta$  (transition function)

$$\langle \langle q, a, E \rangle, \langle q, Y X \rangle \rangle$$

$$\langle \langle q, *, Y \rangle, \langle q, T \rangle \rangle$$

$$\langle \langle q, b, T \rangle, \langle q, \varepsilon \rangle \rangle$$

$$\langle \langle q, +, X \rangle, \langle q, E \rangle \rangle$$

$$\langle \langle q, a, E \rangle, \langle q, \varepsilon \rangle \rangle$$

# Theorem 1

- **Theorem 1.** *If a language is defined by a CFG then it is accepted by a PDA.*
- **Proof.** Let  $z, y \in \Sigma^*$ ,  $\gamma \in N^*$ ,  $A \in N$ .
  - Construct the PDA as outlined earlier.
  - Then  $A \xRightarrow{\text{lm}} z\gamma \Leftrightarrow \langle q, zy, A \rangle \xrightarrow{n} \langle q, y, \gamma \rangle$ .
    - The grammar is on Greibach normal form:
      - Each derivation step and each next-config. step handles one non-terminal
      - Can formally be proved by induction over the number of steps!
  - From this it follows  $S \xRightarrow{*} X \Leftrightarrow \langle q, x, S \rangle \xrightarrow{*} \langle q, \varepsilon, \varepsilon \rangle$ 
    - i.e. if a string can be derived by the CFG it is accepted by the PDA.

# Theorem 1, notes

- The construction presented here especially *eases the induction proof*.
- In general you don't need the grammar to be on Greibach normal form.
- If you just want to do a transformation of a grammar into a PDA you can allow terminals on the stack.
  - For every grammar rule  $A \rightarrow \alpha$ : let  $\langle\langle q, \varepsilon, A \rangle, \langle q, \alpha \rangle\rangle$  be an element in  $\delta$ .
  - For every terminal  $a$ : let  $\langle\langle q, a, a \rangle, \langle q, \varepsilon \rangle\rangle$  be an element in  $\delta$ .

# PDA to CFG

- In this section we prove that given a PDA, we can construct a CFG that describes the language of the PDA.
  - First we transform a one-state PDA to a grammar
  - Then we will convert a many-state one to the one-state form.

# PDA to CFG: A special case

- Given a one-state PDA:  $M = \langle \{q\}, \Sigma, \Gamma, \delta, s, \perp, F \rangle$
- The only important property of the PDA is its single state.
  - $\Sigma$  and  $\Gamma$  may overlap.
  - The state may be final or not.
- For every element  $\langle \langle q, c, A \rangle, \langle q, B_1 B_2 \dots B_n \rangle \rangle \in \delta$ 
  - Let the grammar have the production rule  $A \rightarrow c B_1 B_2 \dots B_n$ .
  - Let the start stack symbol of the PDA be the start symbol of the CFG.
- From the set of production rules, construct  $G = \langle N, \Sigma, P, S \rangle$

# PDA to CFG: General case

- **Goal:** Converting a many-state PDA into a single-state PDA
- Let the starting point be the PDA  $M = \langle Q, \Sigma, \Gamma, \delta, s, \perp, \{t\} \rangle$ 
  - With just one final state
  - (See lecture 9, construction of  $M'$ , for proof on how to convert a PDA with multiple final states to one final state.)
- **Approach:**
  - Encode *state* information in the *stack symbols* instead.
  - Create new stack symbols with labels on a specific form, representing their intended use.

## PDA to CFG: General case, cont.

- If we with  $M$  have  $\langle p, x, A \rangle \xrightarrow{n} M \langle q, \varepsilon, \varepsilon \rangle$  we will with our new  $M'$  have  $\langle *, x, [pAq] \rangle \xrightarrow{n} M' \langle *, \varepsilon, \varepsilon \rangle$ .
  - Note that **[pAq]**, the new stack symbol, is just a "label" (a name).
  - The PDA doesn't look at its parts: it can't recognize them or separate them.
- Now we can define  $M' = \langle \{*\}, \Sigma, \Gamma', \delta', *, [s \perp t], \emptyset \rangle$ , where  $\Gamma' = Q \times \Gamma \times Q$ 
  - $*$  is our new single state, with a unique name.
- For every element  $\langle \langle p, c, A \rangle, \langle q_0, B_1 B_2 \dots B_n \rangle \rangle \in \delta$  we will have  $\langle \langle *, c, [pAq_n] \rangle, \langle *, [q_0 B_1 q_1][q_1 B_2 q_2] \dots [q_{n-1} B_n q_n] \rangle \rangle \in \delta'$  for all  $q_1, q_2, \dots, q_n \in Q$ .
  - There should be an element in  $\delta'$  for every possible combination of  $n$  states in  $M$ .

# PDA to CFG: General case, cont.

- **Lemma 1.** *If a language is accepted by a many-state PDA, it is accepted by a one-state one.*
- **Proof.** Construct  $M'$  as above.
  - Then  $\langle p, x, B_1B_2\dots B_n \rangle \xrightarrow{n} M \langle q_n, \varepsilon, \varepsilon \rangle \Leftrightarrow \langle *, x, [pB_1q_1][q_1B_2q_2] \dots [q_{n-1}pB_1q_n] \rangle \xrightarrow{n} M' \langle *, \varepsilon, \varepsilon \rangle$
  - The stack symbols in  $M'$  are named after each step in  $M$ , so this could formally be proved by induction over  $n$ .
- From this it follows  $\langle s, x, \perp \rangle \xrightarrow{*} M \langle t, \varepsilon, \varepsilon \rangle \Leftrightarrow \langle *, x, [s\perp t] \rangle \xrightarrow{*} M' \langle *, \varepsilon, \varepsilon \rangle$ ,
  - i.e. if a string is accepted by a many-states PDA it is accepted by a one-state one.
- **Theorem 2.** *If a language is accepted by a PDA then it is the language of a CFG.*
- **Proof.** According to Lemma 1, a many-states PDA can be converted to a one-state one. According to the special case before, a one-state PDA can be transformed into a CFG.

# Deterministic PDA

- In a DFA, the transition function must be a *total function* (= defined for all arguments).
  - In any state, for any symbol read, there must be a state to go to.
  - Sometimes we, informally, allow the transition to be a partial function.
- **Definition 1.** A deterministic pushdown automaton, DPDA, is a PDA where  $\delta$  is a *partial function*.
- So, there is for every state  $p$ 
  - either at most one  $\langle\langle p, a, A \rangle, \langle q, \beta \rangle\rangle$  for each symbol "a" or a  $\langle\langle p, \varepsilon, A \rangle, \langle q', \beta' \rangle\rangle$  in  $\delta$ .
- If a DPDA accepts  $x$  with empty stack it can't accept  $xy$  ( $y \neq \varepsilon$ ) with empty stack.
  - *Prefix property:* A string in the accepted language cannot be a proper prefix of another string in the language.

# Coming up soon ...

- The next week:
  - **Wednesday:** Closure properties and pumping lemma for CFLs
- The week after next:
  - **Monday:** LL(1) and LR(0) parsing
  - **Wednesday:** LR(1) parsing

Thanks for today!