ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Prof. Dr. Andreas Podelski
Matthias Heizmann
Christian Schilling

June 4th, 2014

**3. Lecture**
**Computer Science Theory**

# Chapter III – Context-free languages and pushdown automata (pp. 37-69)

## §2 Pumping lemma (pp. 44-47)

We skip this part in the interest of time.

Summary: There is also a pumping lemma for context-free languages. Again, it can be used to prove that a language is *not* context-free. But, as in the regular language case, there are languages which are not context-free, but for which the pumping lemma is not strong enough to show that.

## §3 Pushdown automata (pp. 48-57)

We want to have an automaton model for context-free languages, like we had the finite automata for regular languages. We have already seen that context-free grammars can generate languages such as $L = \{a^n b^n \mid n \in \mathbb{N}\}$. An automaton would need to store the number of $a$'s read so far. Hence we have to add some memory to our $\varepsilon$-NFA model.

For this we use a *stack* (of unlimited size) on which the automaton can write symbols from a new *stack alphabet*. The automaton can *pop* (erase) the topmost symbol and use it for choosing the next transition. Also, it can *push* (add) an arbitrary string to the stack. In each step exactly one pop and one push operation is applied. We call this model a *pushdown automaton* (PDA).

Unlike finite automata, *deterministic PDAs* (DPDAs) are weaker than nondeterministic PDAs, i.e., there are context-free languages for which there is no DPDA accepting it.

Formally, a (*nondeterministic*) *push-down automaton* (PDA) is a 7-tuple

$$\mathcal{A} = (\Sigma, Q, \Gamma, \rightarrow, q_0, Z_0, F)$$

with the following properties:

- $\Sigma$ is the *input alphabet*,
- $Q$ is a finite set of *states*,
- $\Gamma$ is the *stack alphabet*,
- $\rightarrow \subseteq Q \times \Gamma \times (\Sigma \cup \{\varepsilon\}) \times Q \times \Gamma^*$ is the *transition relation*,
- $q_0 \in Q$ is the *initial/start state*,
- $Z_0 \in \Gamma$ is the *initial/start symbol of the stack*,
- $F \subseteq Q$ is the set of *final states*.

Transition are of the form $(q, Z, \alpha, q', \gamma)$. We also write $(q, Z) \xrightarrow{\alpha} (q', \gamma)$.

A PDA reads a word $w$ as follows. Initially, the stack contains only $Z_0$ and the current state is $q_0$. Then in each step the PDA pops the top-most symbol from the stack. It then either reads the next symbol of $w$ or it reads nothing (spontaneous $\varepsilon$-transition). In both cases, depending on the transition relation, it goes to a new state and pushes some new string (possibly $\varepsilon$).

Note: When the stack is empty *after* a step (i.e., the last symbol was popped and no symbol was pushed), the PDA stops working. ✐(1)

### 1.2.1 Acceptance

Since we have a stack now, we can define two different notions of *acceptance*:

(a) We write $L(\mathcal{A})$ for the language accepted with final states (as usual).

(b) We write $L_\varepsilon(\mathcal{A})$ for the language accepted with the empty stack.

It turns out that these acceptance conditions are equally powerful.

**3.5 Theorem (acceptance)**

(a) For every PDA $\mathcal{A}$ we can construct a PDA $\mathcal{B}$ with $L(\mathcal{A}) = L_\varepsilon(B)$.

(b) For every PDA $\mathcal{A}$ we can construct a PDA $\mathcal{B}$ with $L_\varepsilon(\mathcal{A}) = L(B)$.