Theory of Computer Science

C6. Context-free Languages: Push-down Automata

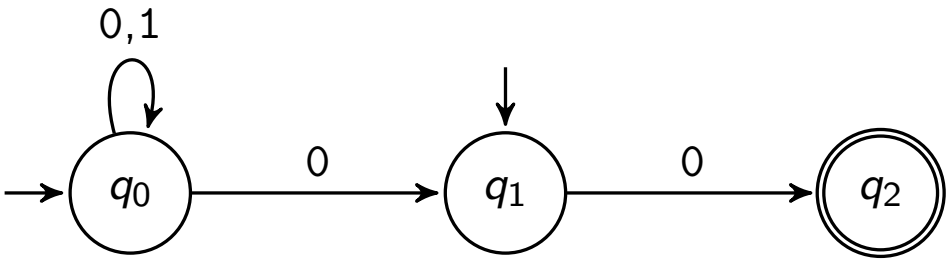
Malte Helmert

University of Basel

April 5, 2017

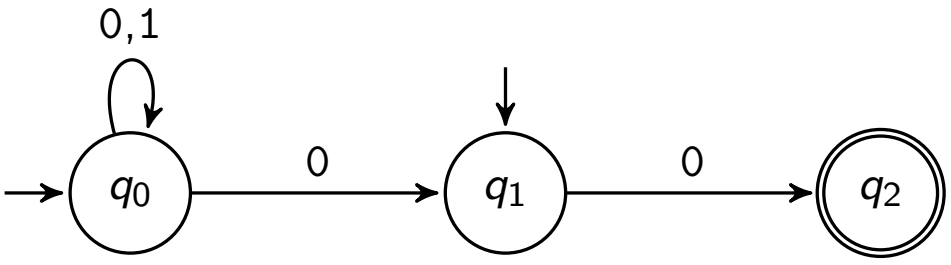
Motivation

Limitations of Finite Automata



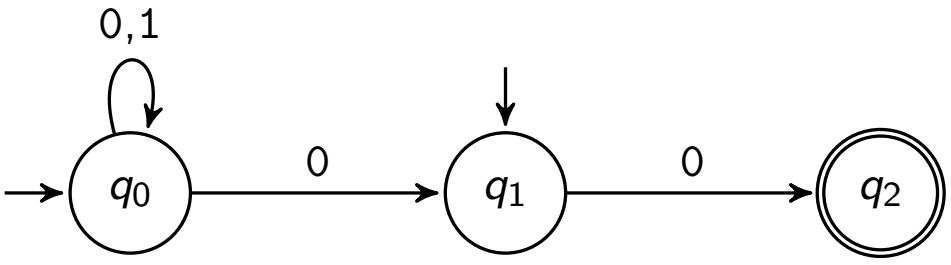
- Language L is regular.
  $\Longleftrightarrow$ There is a finite automaton that accepts L.

Limitations of Finite Automata



- Language L is regular.
  $\Longleftrightarrow$ There is a finite automaton that accepts L.
- What information can a finite automaton "store" about the already read part of the word?

Limitations of Finite Automata



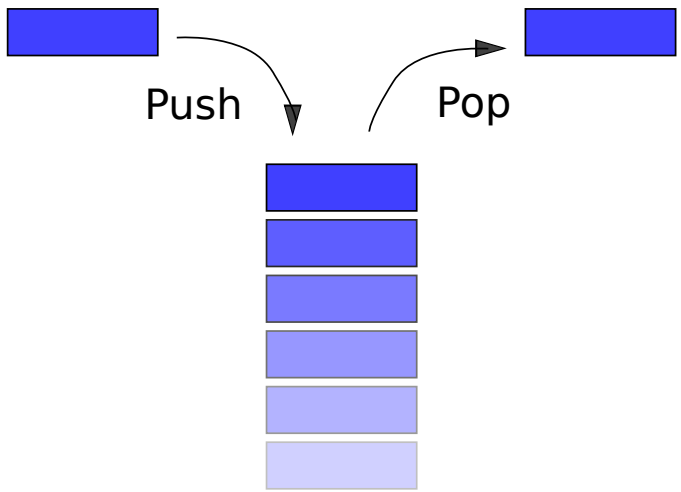
- Language L is regular.
  $\Longleftrightarrow$ There is a finite automaton that accepts L.
- What information can a finite automaton "store" about the already read part of the word?
- Infinite memory would be required for
  $L = \{x_1 x_2 \ldots x_n x_n \ldots x_2 x_1 \mid n > 0, x_i \in \{\mathtt{a}, \mathtt{b}\}\}$.
- therefore: extension of the automata model with memory

Stack

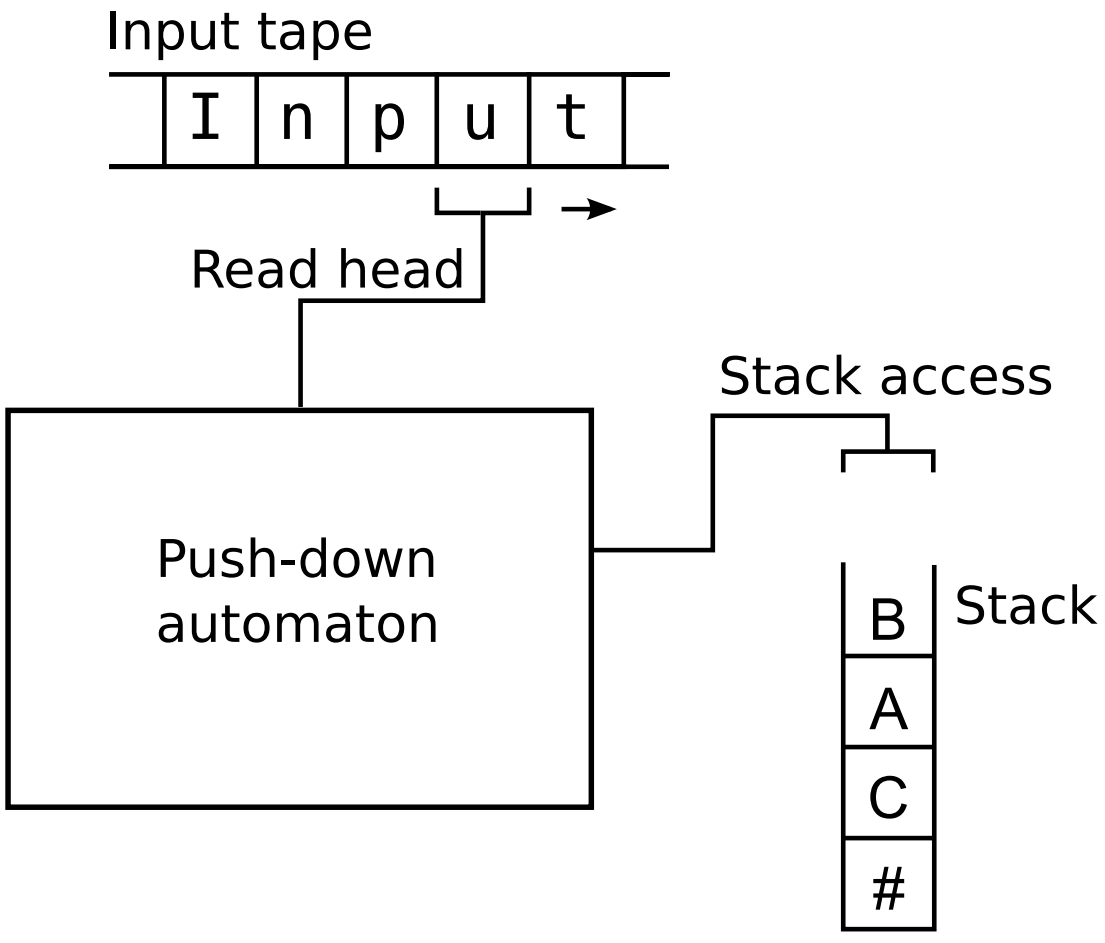
A stack is a data structure following the last-in-first-out (LIFO) principle supporting the following operations:

- push: puts an object on top of the stack
- pop: removes the object at the top of the stack
- peek: returns the top object without removing it

German: Keller, Stapel



Push-down Automata: Visually



German: Kellerautomat, Eingabeband, Lesekopf, Kellerzugriff

Push-Down Automata

Push-down Automata: Definition

Definition (Push-down Automaton)

A push-down automaton (PDA) is a 6-tuple $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \# \rangle$ with

- Q finite set of states
- Σ the input alphabet
- Γ the stack alphabet
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}_\mathrm{f}(Q \times \Gamma^*)$ the transition function (where $\mathcal{P}_\mathrm{f}$ is the set of all finite subsets)
- $q_0 \in Q$ the start state
- $\# \in \Gamma$ the bottommost stack symbol

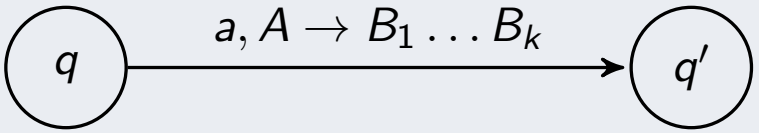
German: Kellerautomat, Eingabealphabet, Kelleralphabet, Überführungsfunktion

Push-down Automata: Transition Function

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \# \rangle$ be a push-down automaton.

What is the Intuitive Meaning of the Transition Function δ?

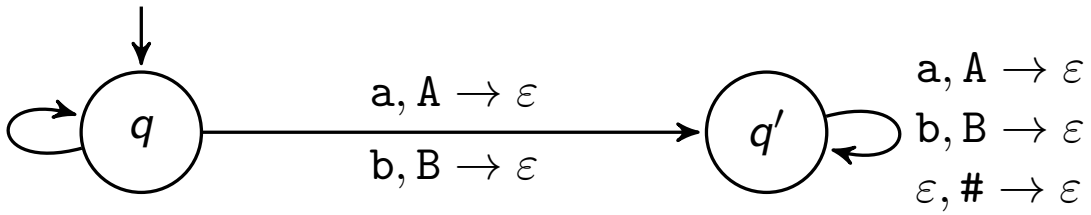
- $\langle q', B_1 \dots B_k \rangle \in \delta(q, a, A)$: If M is in state q, reads symbol a and has A as the topmost stack symbol,
  then M can transition to q' in the next step while replacing A with $B_1 \dots B_k$ (afterwards B_1 is the topmost stack symbol)



- special case $a = \varepsilon$ is allowed (spontaneous transition)

Push-down Automata: Example

$\mathtt{a}, \mathtt{A} \to \mathtt{AA}$
$\mathtt{a}, \mathtt{B} \to \mathtt{AB}$
$\mathtt{a}, \# \to \mathtt{A}\#$
$\mathtt{b}, \mathtt{A} \to \mathtt{BA}$
$\mathtt{b}, \mathtt{B} \to \mathtt{BB}$
$\mathtt{b}, \# \to \mathtt{B}\#$



$M = \langle \{q, q'\}, \{\mathtt{a}, \mathtt{b}\}, \{\mathtt{A}, \mathtt{B}, \#\}, \delta, q, \# \rangle$ with

$$\delta(q, \mathtt{a}, \mathtt{A}) = \{\langle q, \mathtt{AA}\rangle, \langle q', \varepsilon\rangle\} \quad \delta(q, \mathtt{b}, \mathtt{A}) = \{\langle q, \mathtt{BA}\rangle\} \quad \delta(q, \varepsilon, \mathtt{A}) = \emptyset$$
$$\delta(q, \mathtt{a}, \mathtt{B}) = \{\langle q, \mathtt{AB}\rangle\} \quad \delta(q, \mathtt{b}, \mathtt{B}) = \{\langle q, \mathtt{BB}\rangle, \langle q', \varepsilon\rangle\} \quad \delta(q, \varepsilon, \mathtt{B}) = \emptyset$$
$$\delta(q, \mathtt{a}, \#) = \{\langle q, \mathtt{A}\#\rangle\} \quad \delta(q, \mathtt{b}, \#) = \{\langle q, \mathtt{B}\#\rangle\} \quad \delta(q, \varepsilon, \#) = \emptyset$$
$$\delta(q', \mathtt{a}, \mathtt{A}) = \{\langle q', \varepsilon\rangle\} \quad \delta(q', \mathtt{b}, \mathtt{A}) = \emptyset \quad \delta(q', \varepsilon, \mathtt{A}) = \emptyset$$
$$\delta(q', \mathtt{a}, \mathtt{B}) = \emptyset \quad \delta(q', \mathtt{b}, \mathtt{B}) = \{\langle q', \varepsilon\rangle\} \quad \delta(q', \varepsilon, \mathtt{B}) = \emptyset$$
$$\delta(q', \mathtt{a}, \#) = \emptyset \quad \delta(q', \mathtt{b}, \#) = \emptyset \quad \delta(q', \varepsilon, \#) = \{\langle q', \varepsilon\rangle\}$$

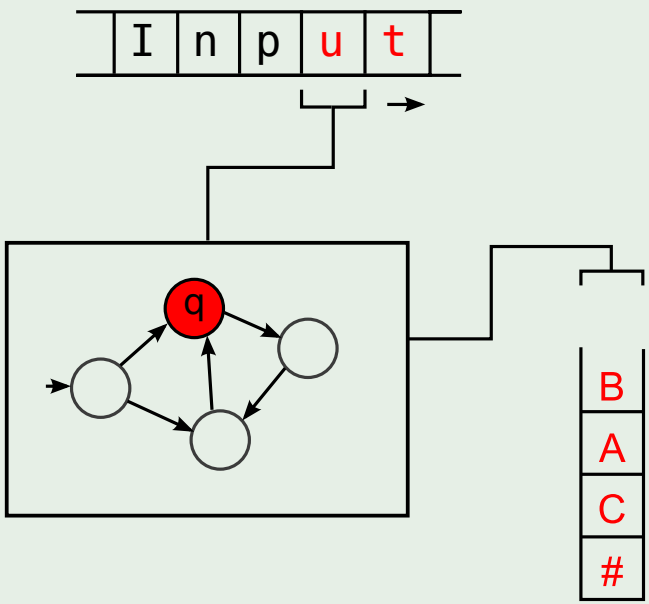
Push-down Automata: Configuration

Definition (Configuration of a Push-down Automaton)

A configuration of a push-down automaton $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \# \rangle$ is given by a triple $c \in Q \times \Sigma^* \times \Gamma^*$.

German: Konfiguration

Example



Configuration $\langle q, \mathtt{ut}, \mathtt{BAC\#} \rangle$.

Push-down Automata: Steps

Definition (Transition/Step of a Push-down Automaton)

We write $c \vdash_M c'$ if a push-down automaton $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \# \rangle$ can transition from configuration c to configuration c' in one step. Exactly the following transitions are possible:

$$\langle q, a_1 \dots a_n, A_1 \dots A_m \rangle \vdash_M \begin{cases} \langle q', a_2 \dots a_n, B_1 \dots B_k A_2 \dots A_m \rangle \\ \qquad \text{if } \langle q', B_1 \dots B_k \rangle \in \delta(q, a_1, A_1) \\ \langle q', a_1 a_2 \dots a_n, B_1 \dots B_k A_2 \dots A_m \rangle \\ \qquad \text{if } \langle q', B_1 \dots B_k \rangle \in \delta(q, \varepsilon, A_1) \end{cases}$$

German: Übergang

If M is clear from context, we only write $c \vdash c'$.

Push-down Automata: Reachability of Configurations

Definition (Reachable Configuration)

Configuration c' is reachable from configuration c in PDA M ($c \vdash_M^* c'$) if there are configurations $c_0, \ldots, c_n$ ($n \geq 0$) where

- $c_0 = c$,
- $c_i \vdash_M c_{i+1}$ for all $i \in \{0, \ldots, n-1\}$, and
- $c_n = c'$.

German: c' ist in M von c erreichbar

Push-down Automata: Recognized Words

Definition (Recognized Word of a Push-down Automaton)

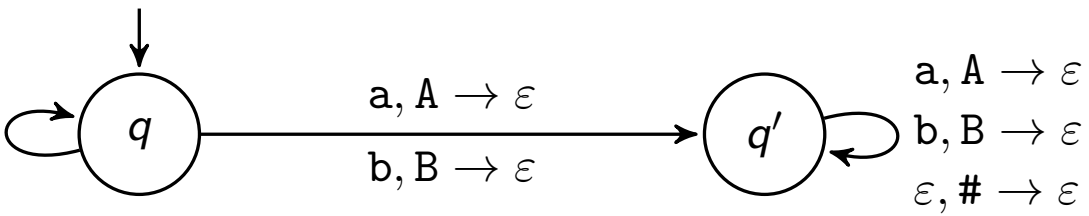
PDA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \# \rangle$ recognizes the word $w = a_1 \dots a_n$ iff the configuration $\langle q, \varepsilon, \varepsilon \rangle$ (word processed and stack empty) for some $q \in Q$ is reachable from the start configuration $\langle q_0, w, \# \rangle$.

$$M \text{ recognizes } w \text{ iff } \langle q_0, w, \# \rangle \vdash_M^* \langle q, \varepsilon, \varepsilon \rangle \text{ for some } q \in Q.$$

German: M erkennt w, Startkonfiguration

Push-down Automata: Recognized Word Example

$\mathtt{a}, \mathtt{A} \to \mathtt{AA}$
$\mathtt{a}, \mathtt{B} \to \mathtt{AB}$
$\mathtt{a}, \# \to \mathtt{A\#}$
$\mathtt{b}, \mathtt{A} \to \mathtt{BA}$
$\mathtt{b}, \mathtt{B} \to \mathtt{BB}$
$\mathtt{b}, \# \to \mathtt{B\#}$



example: this PDA recognizes bbabbabb $\rightsquigarrow$ blackboard

Push-down Automata: Accepted Language

Definition (Accepted Language of a Push-down Automaton)

Let M be a push-down automaton with input alphabet Σ.
The language accepted by M is defined as

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid M \text{ recognizes } w\}.$$

example: blackboard

Questions

Questions?

PDAs vs. Context-free Languages

PDAs Accept Exactly the Context-free Languages

Theorem

A language L is context-free if and only if
L is accepted by a push-down automaton.

PDAs Accept Exactly the Context-free Languages

Proof.

$\Rightarrow$: Let $G = \langle \Sigma, V, P, S \rangle$ be a context-free grammar for L.
The push-down automaton $M = \langle \{q\}, \Sigma, V \cup \Sigma, \delta, q, S \rangle$
with the following δ accepts L:

- $\langle q, w \rangle \in \delta(q, \varepsilon, A)$ for every rule $A \to w \in P$ with $w \in (V \cup \Sigma)^*$.
- $\langle q, \varepsilon \rangle \in \delta(q, a, a)$ for $a \in \Sigma$.

...

PDAs Accept Exactly the Context-free Languages

Proof.

$\Rightarrow$: Let $G = \langle \Sigma, V, P, S \rangle$ be a context-free grammar for L.
The push-down automaton $M = \langle \{q\}, \Sigma, V \cup \Sigma, \delta, q, S \rangle$
with the following δ accepts L:

- $\langle q, w \rangle \in \delta(q, \varepsilon, A)$ for every rule $A \to w \in P$ with $w \in (V \cup \Sigma)^*$.
- $\langle q, \varepsilon \rangle \in \delta(q, a, a)$ for $a \in \Sigma$.

Because:
$x \in \mathcal{L}(G)$
iff there is a derivation of the form $S \Rightarrow \cdots \Rightarrow x$ in G
iff there is a sequence of configurations of M with
$\langle q, x, S \rangle \vdash \cdots \vdash \langle q, \varepsilon, \varepsilon \rangle$
iff $x \in \mathcal{L}(M)$ ...

PDAs Accept Exactly the Context-free Languages

Proof (continued).

$\Leftarrow$: omitted $\square$

Questions

Questions?

Summary

Summary

- Push-down automata (PDAs) extend NFAs with memory.
- PDAs accept not with end states but with an empty stack.
- The languages accepted by PDAs are exactly the context-free languages.

Further Topics on Context-free Languages and PDAs

- With the CYK-algorithm (Cocke, Younger and Kasami) it is possible to decide $w \in \mathcal{L}(G)$ in time $O(|w|^3)$ for a grammar in Chomsky normal form and a word w.
- Deterministic push-down automata have the restriction $|\delta(q, a, A)| + |\delta(q, \varepsilon, A)| \leq 1$ for all $q \in Q, a \in \Sigma, A \in \Gamma$. They accept with end states rather than empty stack.
- The languages accepted by deterministic PDAs are called deterministic context-free languages. They form a strict superset of the regular languages and a strict subset of the context-free languages.