Pushdown automata

H. Geuvers and A. Kissinger

Institute for Computing and Information Sciences – Intelligent Systems
Radboud University Nijmegen

Version: 2015-16

Outline

Automata for Context-Free Languages

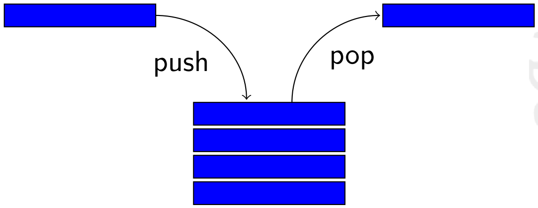
Language class	Syntax/Grammar	Automata
Context-free	context-free grammar	?
Regular	regular expressions, regular grammar	DFA, NFA, NFA_λ

- DFA, NFA, NFA_λ: finite states = finite memory, e.g.
  - even or odd number of a's read: two states *even*, *odd*
  - the last 2 letters read: four states for aa, ab, ba, bb.
- **Problem**: languages like $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$ need unbounded memory. A DFA with k states can only "count to k".
- **Solution:** extend NFA_λ by adding memory

Automata for Context-Free Languages

Various simple memory models are possible:

- Queue: First in, first out (like waiting in line, see: the English)
- Stack: Last in, first out (like a laundry basket)



Stack Memory

A pushdown automaton is an NFA_λ with a stack.

A stack can be described as a word over the stack alphabet Γ:

- Empty stack is λ.
- $\text{push}(X, YZZY) = XYZZY$, push a new element on top (note top=left).
- $\text{pop}(YZZY) = ZZY$, remove the top element.
- $\text{top}(YZZY) = Y$, look at the top element.

Note: the empty stack has no top.

Pushdown Automaton

Def. A **pushdown automaton (PDA)** $M = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$ consists of

- a finite set of states Q
- an input alphabet Σ
- an initial state $q_0 \in Q$
- a set of final states $F \subseteq Q$
- **a stack alphabet** Γ
- **a transition function** $\delta \colon Q \times \Sigma_\lambda \times \Gamma_\lambda \to \mathcal{P}(Q \times \Gamma_\lambda)$ where $\Sigma_\lambda = \Sigma \cup \{\lambda\}$ and $\Gamma_\lambda = \Gamma \cup \{\lambda\}$.

Pushdown Automaton

- An NFA$_\lambda$ transition looks like this:

$$q \in \delta(p, \mathtt{a}) \quad \leftrightarrow \quad p \xrightarrow{\mathtt{a}} q$$

- A PDA adds (optional) stack elements to pop and push:

$$\langle q, B \rangle \in \delta(p, \mathtt{a}, A) \quad \leftrightarrow \quad p \xrightarrow{\mathtt{a},\, A/B} q \qquad p \xrightarrow{\overset{\text{INPUT}}{\mathtt{a}},\, \overset{\text{POP}}{A}/\overset{\text{PU}}{B}} q$$

Pushdown Automaton Computation

A computation of a PDA M on input word w:

- Start configuration $\langle q_0, w, \lambda\rangle$
  (start in initial state with empty stack)
- Transitions are taken (nondeterministically) depending on
  – the next input symbol (as in DFA, NFA) and
  – the stack top symbol.
  Changes configuration $\langle q, w, \alpha\rangle \to \langle q', w', \alpha'\rangle$ according to transition function
- Computation is successful if it ends in a configuration $\langle q, \lambda, \lambda\rangle$ where $q \in F$.
  (Acceptance by final state and empty stack)

Language Accepted by a PDA

Def. The language accepted by a PDA M is:

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid \langle q_0, w, \lambda \rangle \Rightarrow \langle q, \lambda, \lambda \rangle, q \in F\}.$$

....where $\langle q_0, w, \lambda \rangle \Rightarrow \langle q, \lambda, \lambda \rangle$ means:

$$\langle q_0, w, \lambda \rangle \to \langle q, w', \alpha \rangle \to \cdots \to \langle q, \lambda, \lambda \rangle$$

The stack **starts empty** and must **finish empty**.

Pushdown Automaton Transitions

normal transition: $\quad p \xrightarrow{\mathrm{a},\, A/B} q$

- **can be taken if:** next input symbol is a, stack top is A
- **actions:** read a from word, pop A, push B

Pushdown Automaton Transitions

normal transition: $p \xrightarrow{\mathsf{a},\, A/B} q$

only pop: $p \xrightarrow{\mathsf{a},\, A/\lambda} q$

- **can be taken if:** next input symbol is a, stack top is A
- **actions:** read a from word, pop A, ~~push B~~

Pushdown Automaton Transitions

normal transition: $p \xrightarrow{\mathsf{a},\, A/B} q$

only pop: $p \xrightarrow{\mathsf{a},\, A/\lambda} q$

only push: $p \xrightarrow{\mathsf{a},\, \lambda/B} q$

- **can be taken if:** next input symbol is a, ~~stack top is A~~
- **actions:** read a from word, ~~pop A~~, push B

Pushdown Automaton Transitions (cont'd)

ignore stack: $$p \xrightarrow{a,\ \lambda/\lambda} q$$

- **can be taken if:** next input symbol is a
- **actions:** read a from word

Pushdown Automaton Transitions (cont'd)

ignore stack: $p \xrightarrow{\mathsf{a},\, \lambda/\lambda} q$

ignore word: $p \xrightarrow{\lambda,\, A/B} q$

- **can be taken if:** stack top is A
- **actions:** pop A, push B

Pushdown Automaton Transitions (cont'd)

ignore stack: $p \xrightarrow{a,\,\lambda/\lambda} q$

ignore word: $p \xrightarrow{\lambda,\,A/B} q$

ignore everything: $p \xrightarrow{\lambda,\,\lambda/\lambda} q$

- **can be taken:** any time
- **actions:** do nothing

Pushdown Automaton Transitions (cont'd)

ignore stack: $p \xrightarrow{a,\,\lambda/\lambda} q$

ignore word: $p \xrightarrow{\lambda,\,A/B} q$

ignore everything: $p \xrightarrow{\lambda,\,\lambda/\lambda} q$

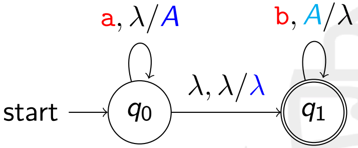
$p \xrightarrow{\lambda,\,\lambda/B} q$ $\qquad$ $p \xrightarrow{\lambda,\,A/\lambda} q$

PDA example 1

$L = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\} \ni \lambda, \mathtt{ab}, \mathtt{aabb}, \ldots$

Idea: count the a's.

Take $\Sigma = \{\mathtt{a}, \mathtt{b}\}, \Gamma = \{A\}$,



Example computations...

- $\langle q_0, \lambda, \lambda\rangle \to \langle q_1, \lambda, \lambda\rangle$✓ (SUCCESS)
- $\langle q_0, \underline{\mathtt{a}}\mathtt{abb}, \lambda\rangle \to \langle q_0, \underline{\mathtt{a}}\mathtt{bb}, A\rangle \to \langle q_0, \underline{\mathtt{b}}\mathtt{b}, AA\rangle \to$
  $\langle q_1, \underline{\mathtt{b}}\mathtt{b}, AA\rangle \to \langle q_1, \underline{\mathtt{b}}, A\rangle \to \langle q_1, \lambda, \lambda\rangle$✓ (SUCCESS)
- $\langle q_0, \underline{\mathtt{a}}\mathtt{ba}, \lambda\rangle \to \langle q_0, \underline{\mathtt{b}}\mathtt{a}, A\rangle \to \langle q_1, \underline{\mathtt{b}}\mathtt{a}, A\rangle \to \langle q_1, \underline{\mathtt{a}}, \lambda\rangle$☠ (STUCK)
  $\langle q_0, \underline{\mathtt{a}}\mathtt{ba}, \lambda\rangle \to \langle q_1, \mathtt{aba}, \lambda\rangle$☠ (STUCK)

Remarks on PDA Transitions

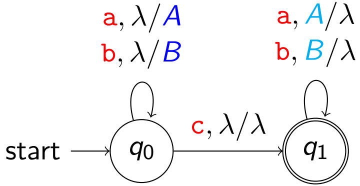
- Note: $p \xrightarrow{a,\, \lambda / B} q$ does not mean that the stack must be empty to take the transition.
- Note: $p \xrightarrow{a,\, A / \lambda} q$ does not mean that the stack is empty after the transition.

PDA example 2

$$L = \{w\mathtt{c}w^R \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\} \ni \mathtt{c}, \mathtt{abcba}, \mathtt{bbcbb}.$$

Can we find a PDA that accepts L?
Idea: Memorise w-part using the stack, and use it to check for w^R.

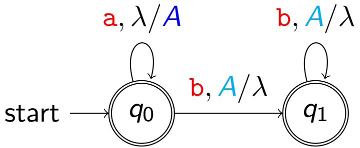
Take: $\Gamma = \{A, B\}$, and Q, δ, F as follows:



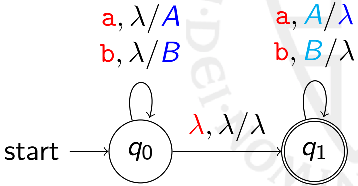
Variations on PDAs

- Acceptance criteria:
  - Acceptance by final state and empty stack (our def.)
  - Acceptance by final state (only)
  - Acceptance by empty stack (only)

  All are equivalent (accept same class of languages).
- A PDA is deterministic if for any combination of state, input symbol and stack top, there is at most one transition possible.



deterministic



nondeterministic

Deterministic Context-Free Languages

Def. A language is called deterministic context-free if there exists a deterministic PDA M with $L = \mathcal{L}(M)$.

Examples of deterministic CFLs:

- $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$
- $\{w\mathtt{c}w^R \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$

Examples of CFLs that are not deterministic:

- $\{ww^R \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$
- $\{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w = w^r\}$ (palindromes)

Note: L is deterministic CFL implies L is unambiguous, but there are unambiguous CFLs that are not deterministic (e.g., palindromes).

Context-Free Languages and PDAs

Last lecture: From regular grammar G to NFA$_\lambda$ M_G.
Now: From context-free grammar G to PDA M_G.

Pushing words

- First, some notation (multiple push):

$$p \xrightarrow{a,\ A/BC} q \quad := \quad p \xrightarrow{a,\ A/C} p' \xrightarrow{\lambda,\ \lambda/B} q$$

$$\langle av, p, Aw\rangle \to \langle v, p, BCw\rangle$$

- ...which generalises to words:

$$p \xrightarrow{a,\ A/w} q \quad := \quad \text{(push } w \text{ in reverse order)}$$

$$\langle av, p, Aw'\rangle \to \langle v, p, ww'\rangle$$

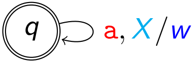
From CFG to PDA

Ideas:

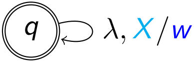
- Put non-terminals on the stack, $\Gamma = V$
- Ensure that rules are of the form $X \to \mathtt{a}w$ or $X \to w$, with $w \in V^*$. This can always be done.
- Use one interesting, accepting, state q, plus more for pushing.
- $\langle q, w, v \rangle \Rightarrow \langle q, \lambda, \lambda \rangle$ in the PDA iff $v \Rightarrow w$ in the CFG

From CFG to PDA (cont.)

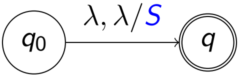
For each production rule $X \to \mathtt{a}w$, add



For each production rule $X \to w$, add



Initially push S onto the stack,



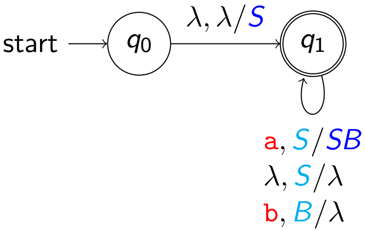
From CFG to PDA, example

$$\boxed{S \;\rightarrow\; aSb \mid \lambda}.$$

This is not of the right form (because of the b), change it to

$$\boxed{\begin{array}{lll} S & \rightarrow & aSB \mid \lambda \\ B & \rightarrow & b \end{array}}.$$

This gives the PDA



From a PDA to a CFG

Ideas:

- Each push of X must have a matching pop of X.
- Split pushes and pops:

$$\text{p} \xrightarrow{\text{a},\, X/Y} \text{q} \quad \Rightarrow \quad \text{p} \xrightarrow{\text{a},\, X/\lambda} \text{p'} \xrightarrow{\lambda,\, \lambda/Y} \text{q}$$

- Just three types of transitions remain:

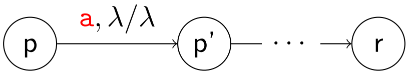
$$p \xrightarrow{\text{a},\, \lambda/X} q, \quad p \xrightarrow{\text{a},\, X/\lambda} q, \quad p \xrightarrow{\text{a},\, \lambda/\lambda} q.$$

(where a can be λ, and p might equal q).

- Take $V = (Q \times Q) \cup \{S\}$.
- $(p, r) \Rightarrow w$ iff $\langle p, w, \lambda \rangle \Rightarrow \langle r, \lambda, \lambda \rangle$.

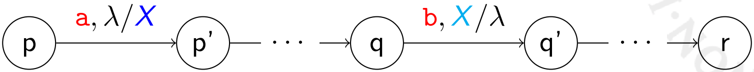
From a PDA to a CFG (cont.)

How can $\langle p, w, \lambda \rangle \Rightarrow \langle r, \lambda, \lambda \rangle$? Either:



new production: $(p, r) \to a(p', r)$

or:



new production: $(p, r) \to a(p', q)b(q', r)$

From a PDA to a CFG (cont.)

Production rules, for pairs of states (p, r):

$$(p, r) \rightarrow a(p', r) \quad \text{for every} \quad p \xrightarrow{a,\ \lambda/\lambda} p'$$

$$(p, r) \rightarrow a(p', q)b(q', r) \quad \text{for every } X, \quad p \xrightarrow{a,\ \lambda/X} p', \quad \text{and} \quad q \xrightarrow{b,\ X/\lambda} q'$$

$$(q, q) \rightarrow \lambda \quad \text{for every } q \in Q$$

$$S \rightarrow (q_0, q) \quad \text{for every } q \in F$$

Beyond context-free languages

One stack ≈ "PDAs can only remember one thing at a time"

Examples of languages that are not context-free:

- $\{a^n b^m a^n b^m \mid n, m \geq 0\}$
- $\{a^n b^n c^n \mid n \geq 0\}$
- $\{w \in \{a, b\}^* \mid |w| \text{ is prime number}\}$

(Prove using pumping lemma for CFLs, Sudkamp 7.4)

Closure properties of context-free languages

If L_1 and L_2 are context-free languages,
then so are:

- $L_1 \cup L_2$ (union),
- $L_1 L_2$ (concatenation),
- L_1^* (star),
- L_1^R (reversal)

but, in general, NOT

- $\overline{L_1}$ (complement),
- $L_1 \cap L_2$ (intersection).

Deterministic CFLs are closed under complement, but NOT under union.

Questions about context-free languages

Decidable (general algorithm exists)

- Given any CFG G and $w \in \Sigma^*$, is $w \in \mathcal{L}(G)$? (build PDA).
- Given PDA M, is $\mathcal{L}(M) = \emptyset$?
- Given PDA M, is $\mathcal{L}(M)$ finite?
- Given any CFG G, is $\mathcal{L}(G)$ regular?

Undecidable (no general algorithm exists)

- Given any CFG G, is $\mathcal{L}(G) = \Sigma^*$?
- Given any CFGs G_1 and G_2, is $\mathcal{L}(G_1) = \mathcal{L}(G_2)$?
- Given any CFG G, is $\mathcal{L}(G)$ deterministic?
- Given any CFG G, is it ambiguous?

Summary

- Context-free grammars generate context-free languages.
- All regular languages are context-free, but not vice versa.
- Context-free languages are accepted by PDAs.
- PDAs cannot be determinised (no subset construction)