Pushdown automata

Twan van Laarhoven

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

Version: fall 2014

Outline

Automata for Context-Free Languages

Language class	**Syntax/Grammar**	**Automata**
Context-free	context-free grammar	?
Regular	regular expressions, regular grammar	DFA, NFA, NFA$_\lambda$

- DFA, NFA, NFA$_\lambda$: finite memory, for example
  - even or odd number of a's read: two states *even*, *odd*
  - the last 2 letters read: four states for *aa*, *ab*, *ba*, *bb*.
- For example: $\{a^n b^n \mid n \geq 0\}$ not regular. We would need to remember how many a's we have seen. A DFA with k states can only "count to k".

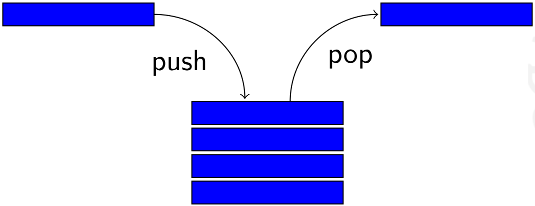
We need some form of unbounded memory.

Automata for Context-Free Languages

The limitation of DFAs is that they dont have a memory.

Various (simple) memory models are possible:

- Queue: First in, first out (like a plastic cup dispenser in a coffee machine)
- Stack: Last in, first out (like plates in a student restaurant)



Pushdown automata extend automata with a stack

Note: only top of stack is visible.

Stack Memory

Pushdown automata extend automata with a stack.

Each item carries an element of stack alphabet Γ.

Stack can be described as a word over Γ, with top $=$ left:

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

Note: the empty stack has no top.

Pushdown Automaton

A pushdown automaton is an NFA_λ equipped with a stack.

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 $\Gamma_\lambda = \Gamma \cup \{\lambda\}$.
  That is, for $p \in Q, a \in \Sigma_\lambda, A \in \Gamma_\lambda$, $\delta(p, a, A)$ is a set of pairs $\langle q, B \rangle$ where $q \in Q, B \in \Gamma_\lambda$ (nondeterministic).

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. (Details next slide.)
  Changes configuration $\langle p, u, \alpha \rangle \to \langle q, v, \beta \rangle$ according to transition function (where $p, q \in Q, ; u, v \in \Sigma^*, \ \alpha, \beta \in \Gamma^*$).
- 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)

Pushdown Automaton Transitions

$\langle q, B\rangle \in \delta(p, \mathsf{a}, A)$ i.e. $p \xrightarrow{\mathsf{a},\, A/B} q$

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

$\langle q, B\rangle \in \delta(p, \lambda, A)$ i.e. $p \xrightarrow{\lambda,\, A/B} q$ (input λ-transition):

- can be taken if: stack top is A (regardless of next input)
- actions: pop A from stack, push B.

$\langle q, B\rangle \in \delta(p, \mathsf{a}, B)$ i.e. $p \xrightarrow{\lambda,\, \lambda/B} q$

- can be taken if: next input is a (regardless of stack top)
- actions: read/consume a, push B.

Pushdown Automaton Transitions (cont.)

Transitions of the form $\langle q, \lambda \rangle \in \delta(p, -, -)$

- can be taken if: as before.
- actions: as before, except nothing is pushed onto stack.

For example:

$\langle q, \lambda \rangle \in \delta(p, \mathtt{a}, \lambda)$ i.e. $p \xrightarrow{\mathtt{a}, \lambda/\lambda} q$

- can be taken if: next input symbol is a
- actions: read/consume a.

$\langle q, \lambda \rangle \in \delta(p, \lambda, \lambda)$ i.e. $p \xrightarrow{\lambda, \lambda/\lambda} q$

- Can be taken: always
- Actions: nothing.

Language Accepted by a PDA

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, F\rangle$ be a PDA, and
let $p, q \in Q,\; u, v \in \Sigma^*,\; \alpha, \beta \in \Gamma^*$.

Notation: $(p, u, \alpha) \Rightarrow (q, v, \beta)$ if there exists a sequence of transitions that change $\langle p, u, \alpha\rangle$ into $\langle q, v, \beta\rangle$.

Def. The language accepted by M is:

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

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.
- Shorthand notation (multiple push):

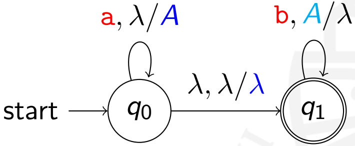
$$p \xrightarrow{a, A/BC} q \quad = \quad p \xrightarrow{a, A/C} p' \xrightarrow{\lambda, \lambda/B} q$$

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:

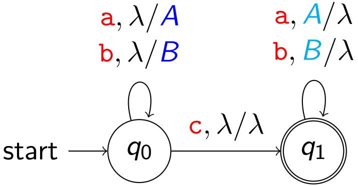
- $\langle q_0, \lambda, \lambda\rangle \to \langle q_1, \lambda, \lambda\rangle$✓ (SUCCESS)
- $\langle q_0, \underline{\mathtt{a}}\mathtt{abb}, \lambda\rangle \to \langle q_0, \underline{\mathtt{a}}\mathtt{bb}, A\rangle \to \langle q_0, \underline{\mathtt{b}}\mathtt{b}, AA\rangle \to$
  $\langle q_1,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

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:

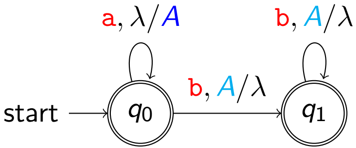


Variations on PDAs

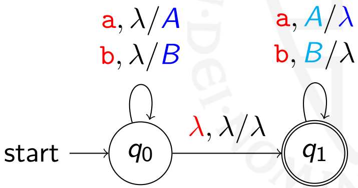
- Acceptance criteria:
  - Acceptance by final state and empty stack (our def.)
  - Acceptance by final state (only)
  - Acceptance by empty stack (only)

  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
(guess middle of string)

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

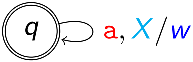
Last lecture: From regular grammar G to NFA$_\lambda$ M_G.
Now: From context-free grammar G to PDA M_G.

Ideas:

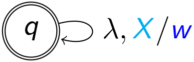
- Put non-terminals on the stack, $\Gamma = V$
- Ensure that rules are of the form $X \to \mathtt{a}w$ or $X \to w$, with $w \in V^*$. This can always be done.
- Use one interesting, accepting, state q, plus more for pushing.
- $\langle q, w, v\rangle \Rightarrow \langle q, \lambda, \lambda\rangle$ in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

Context-Free Languages and PDAs (cont.)

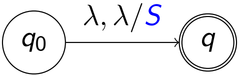
For each production rule $X \to aw$, add



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



Initially push S onto the stack,



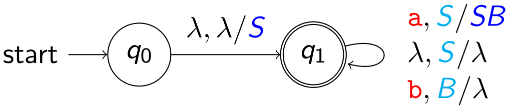
From CFG to PDA, example

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

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

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

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:

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

- Just three types of transitions remain:

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

  (where a can be λ, and p might equal q).
- Take $V = Q \times Q \cup \{S\}$.
- $(p, q) \Rightarrow w$ iff $\langle p, w, \lambda\rangle \Rightarrow \langle q, \lambda, \lambda\rangle$.

From a PDA to a CFG (cont.)

Production rules:

(p, r)	$\to$	$\mathtt{a}(q, r)$	for every $p \xrightarrow{\mathtt{a},\lambda/\lambda} q$
(p, t)	$\to$	$\mathtt{a}(q, r)\mathtt{b}(s, t)$	for every $p \xrightarrow{\mathtt{a},\lambda/X} q$ and $r \xrightarrow{\mathtt{b},X/\lambda} s$
(q, q)	$\to$	λ	for every $q \in Q$
S	$\to$	(q_0, q)	for every $q \in F$

Beyond context-free languages

Examples of languages that are not context-free:

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

(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).

Let S_1 and S_2 be start symbols in grammars G_1 and G_2.
Make new grammar with start symbol S:

union:	$S \to S_1 \mid S_2$,	concatenation:	$S \to S_1 S_2$
star:	$S \to S_1 S \mid \lambda$,	reversal:	(exercise)

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)