COMPSCI 501: Formal Language Theory

Lecture 9: Equivalence of CFG and PDA

Marius Minea
marius@cs.umass.edu

University of Massachusetts Amherst

11 February 2019

CFG and PDA equivalence

Theorem: A language is context-free if and only if some pushdown automaton recognizes it

- Two constructions, one for each direction
- CFG → PDA: nondeterminism needed to choose rule
- PDA → CFG: one rule per state pair, induction proof

Example: recognizing a CFL

Prefix expressions:

$E \to num \mid +\ E\ E \mid *\ E\ E$

top-down

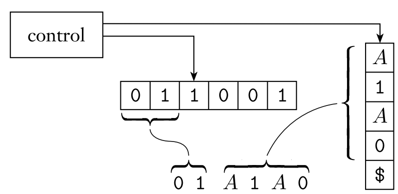
Postfix expressions

$E \to num \mid E\ E\ + \mid E\ E\ *$

top-down
bottom-up

Constructing a PDA from a CFG

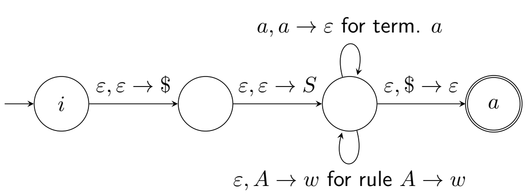
PDA successively applies production rules



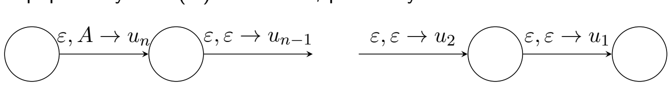
- Initially: start symbol on stack
- Invariant: nonterminal on stack top
- Replace nonterminal with RHS of production rule
  chosen nondeterministically, push symbols right to left
- Match terminal(s) on top of stack with input
  or fail on current branch for chosen rule
- Repeat with top nonterminal until stack empty
  accept if all input read

Construction Details

Overall construction



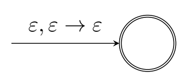
Implementing production rule $A \to u_1 u_2 \dots u_n$:
pop one symbol (A) from stack, push n symbols



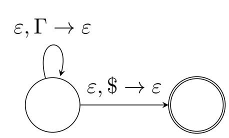
From PDA to CFG

Transform PDA to simplified form

- Single accept state



- Empty stack before accepting



- Each transition either pushes or pops, not both

replace $q_s \xrightarrow{a,b/c} q_f$ with $q_s \xrightarrow{a,b/\varepsilon} q_i \xrightarrow{\varepsilon,\varepsilon/c} q_f$

replace $q_s \xrightarrow{a,\varepsilon/\varepsilon} q_f$ with $q_s \xrightarrow{a,\varepsilon/b} q_i \xrightarrow{\varepsilon,b/\varepsilon} q_f$ for some $b \in \Gamma$

PDA to CFG: State Pair to Nonterminal

Recall NFA → GNFA → regex:
express (regex) all strings taking automaton between two states

Idea: characterize all strings that take automaton from $(p,$ empty stack) to $(q,$ empty stack).
then: same for arbitrary stack, will not be touched

Two cases:

- will empty stack somewhere in between (some state r)
  can express as $A_{pq} \to A_{pr}A_{rq}$
- stack stays nonempty: first: push u, last: pop u (same)

1. push u on input a: $p \xrightarrow{a,\varepsilon/u} r$
2. $r \rightsquigarrow s$ without touching stack (A_{rs})
3. pop u on input b: $s \xrightarrow{b,u/\varepsilon} q$

⇒ add rule $A_{pq} \to aA_{rs}b$

PDA to CFG Construction

Consider PDA $P = (Q, \Sigma, \Gamma, q_0, \{q_{acc}\})$.

Construct grammar G with variables $A_{pq} | p, q \in Q$

1. Add rule $A_{pq} \to aA_{rs}b$ if $p \xrightarrow{a/\text{push } u} r$ and $s \xrightarrow{b/\text{pop } u} q$.
2. Add rule $A_{pq} \to A_{pr}A_{rq}$ for any $p, q, r \in Q$.
3. Add rule $A_{pp} \to \varepsilon$ for any $p \in Q$.
   (no action = empty production)

Start symbol is $A_{q_0,q_{acc}}$.

PDA to CFG Proof (1)

If A_{pq} generates x, then x can bring P from state p with empty stack to state q with empty stack.

Induction by number of steps in derivation $A_{pq} \overset{*}{\Rightarrow} x$

Base case: $k = 1$ step ⇒ RHS has no variables. Only rules are $A_{pp} \to \varepsilon$.
Clearly, with no action, P stays in same state with empty stack.

Inductive step: Assume $A_{pq} \overset{*}{\Rightarrow} x$ in $k + 1$ steps.

Case 1: $A_{pq} \Rightarrow aA_{rs}b$. Then $x = ayb$ and $A_{rs} \overset{*}{\Rightarrow} y$ in k steps.
Thus, P goes on y from r to s on empty stack.
By construction, $p \xrightarrow{a/\text{push } u} r$ and $s \xrightarrow{b/\text{pop } u} q$ for some symbol u, thus we go from p to r to s to q on empty stack. .

Case 2: $A_{pq} \Rightarrow A_{pr}A_{rq}$.
Then $x = yz$ with $A_{pr} \overset{*}{\Rightarrow} y$ and $A_{rq} \overset{*}{\Rightarrow} z$ in $\leq k$ steps.
Thus P goes from p to r by y on empty stack, then from r to q by z on empty stack.

PDA to CFG Proof (2)

If x can bring P from state p with empty stack to state q with empty stack, then A_{pq} generates x.

Induction by number of steps in P' computation $p \rightsquigarrow p$

Base case: $k = 0$. P does nothing reads no input) ⇒ $x = \varepsilon$
This is achieved by rule $A_{pp} \to \varepsilon$

Inductive step: Consider computation of length $k + 1 > 0$.

Case 1: stack only empty at start and end ⇒ must start/end with push u / pop u. Then we have $p \xrightarrow{a/\text{push } u} r$ and $s \xrightarrow{b/\text{pop } u} q$, $x = ayb$.
String y brings P from r to s without touching stack (not emptied).
y takes $k - 1$ steps, so $A_{rs} \overset{*}{\Rightarrow} y$, thus $A_{pq} \overset{*}{\Rightarrow} ayb = x$

Case 2: stack becomes empty at some state r.
$p \rightsquigarrow r$ and $r \rightsquigarrow q$ have each $\leq k$ steps.
Then we have $A_{pr} \overset{*}{\Rightarrow} y$ and $A_{rq} \overset{*}{\Rightarrow} z$
Since we have rule $A_{pq} \to A_{pr}A_{rq}$ we are done.