CS 133 : Automata Theory and Computability

Lecture Slides

Context-Free Grammars and Languages

Francis George C. Cabarle

Algorithms and Complexity Laboratory
Department of Computer Science
University of the Philippines Diliman
fccabarle@up.edu.ph

Day 13

Context-Free Grammars and Languages

Equivalence of PDAs and CFGs

Context-Free Grammars and Languages

Equivalence of PDAs and CFGs

Despite the seemingly different semantics of PDAs (i.e. accepting strings) and CFGs (i.e. generating strings) we show another rather counter-intuitive and nice result: the two are one and the same!

Despite the seemingly different semantics of PDAs (i.e. accepting strings) and CFGs (i.e. generating strings) we show another rather counter-intuitive and nice result: the two are one and the same!

More precisely, either of the two can simulate the other.

What is the point?

Despite the seemingly different semantics of PDAs (i.e. accepting strings) and CFGs (i.e. generating strings) we show another rather counter-intuitive and nice result: the two are one and the same!

More precisely, either of the two can simulate the other.

What is the point?

One, we have another way to characterize (i.e. talk about) context-free languages aside from CFGs.

Despite the seemingly different semantics of PDAs (i.e. accepting strings) and CFGs (i.e. generating strings) we show another rather counter-intuitive and nice result: the two are one and the same!

More precisely, either of the two can simulate the other.

What is the point?

One, we have another way to characterize (i.e. talk about) context-free languages aside from CFGs.

Two, as in a similar case with determinism and nondeterminism, some problems (i.e. languages) may be more easily described with a CFG instead of a PDA, and vice versa.

Theorem

A language is context free if and only if some pushdown automaton recognizes it.

We use notation $(r, u) \in \delta(q, a, s)$ to mean that we change state from q to r on input a, popping s and pushing **string** u onto the stack.

e.g. $\delta(q, a, s) \to (r, xyz)$

Theorem

A language is context free if and only if some pushdown automaton recognizes it.

We use notation $(r, u) \in \delta(q, a, s)$ to mean that we change state from q to r on input a, popping s and pushing **string** u onto the stack.

e.g. $\delta(q, a, s) \to (r, xyz)$ can be written as
$\delta(q, a, s) \to (q_1, z), \delta(q_1, \varepsilon, \varepsilon) \to (q_2, y), \delta(q_2, \varepsilon, \varepsilon) \to (r, x)$.

if part: Convert CFG G to PDA P

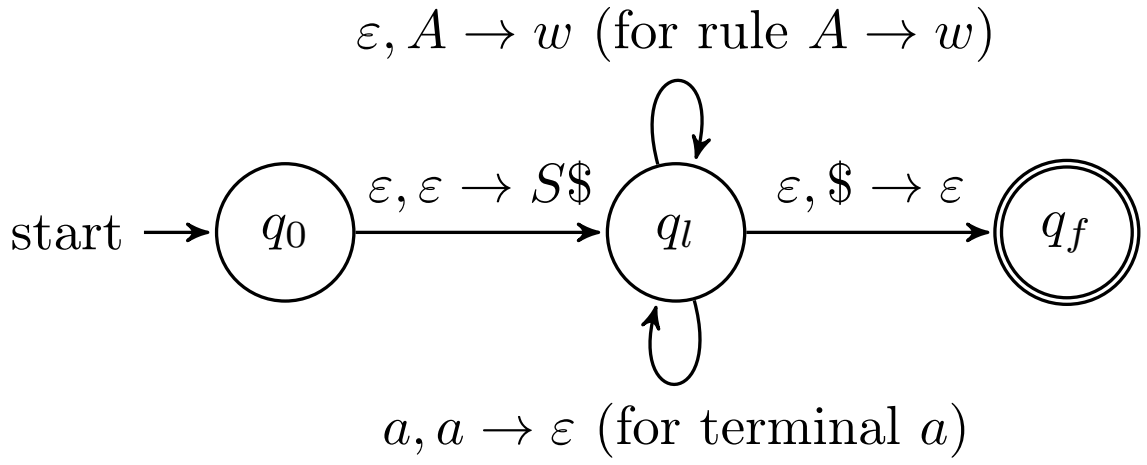
1. Place the marker symbol \$ and the axiom on the stack.
2. Repeat the following steps forever.
   a. If top of stack is a variable A, nondeterministically select one of the rules for A and substitute A by the string on the RHS of the rule.
   b. If top of stack is a terminal a, read next symbol from input and compare it to a. If they match, repeat. If they do not match, reject on this nondeterministic branch.
   c. If top of stack is \$, accept. Doing so accepts the input if it has all been read.

i.e.

if part: Convert CFG G to PDA P

1. Place the marker symbol \$ and the axiom on the stack.
2. Repeat the following steps forever.
   a. If top of stack is a variable A, nondeterministically select one of the rules for A and substitute A by the string on the RHS of the rule.
   b. If top of stack is a terminal a, read next symbol from input and compare it to a. If they match, repeat. If they do not match, reject on this nondeterministic branch.
   c. If top of stack is \$, accept. Doing so accepts the input if it has all been read.

i.e.



An Example

Construct a PDA from the following CFG:

$$S \to aTb \mid b$$
$$T \to Ta \mid \varepsilon$$

An Example

Construct a PDA from the following CFG:

$$S \to aTb \mid b$$
$$T \to Ta \mid \varepsilon$$

only if part: Convert PDA P to CFG G

First, we slightly modify P to give it the following features:

1. It has a *single* accept state, q_f.
2. It *empties* its stack before accepting.
3. Each transition either pushes a symbol onto or pops one off the stack, but NOT both at the same time. (Note: This applies to ε-moves, i.e. if P has an ε-move, modify P such that some "*dummy*" symbol (not in Σ) is pushed onto the stack and is popped off the stack immediately after.)

only if part: Convert PDA P to CFG G

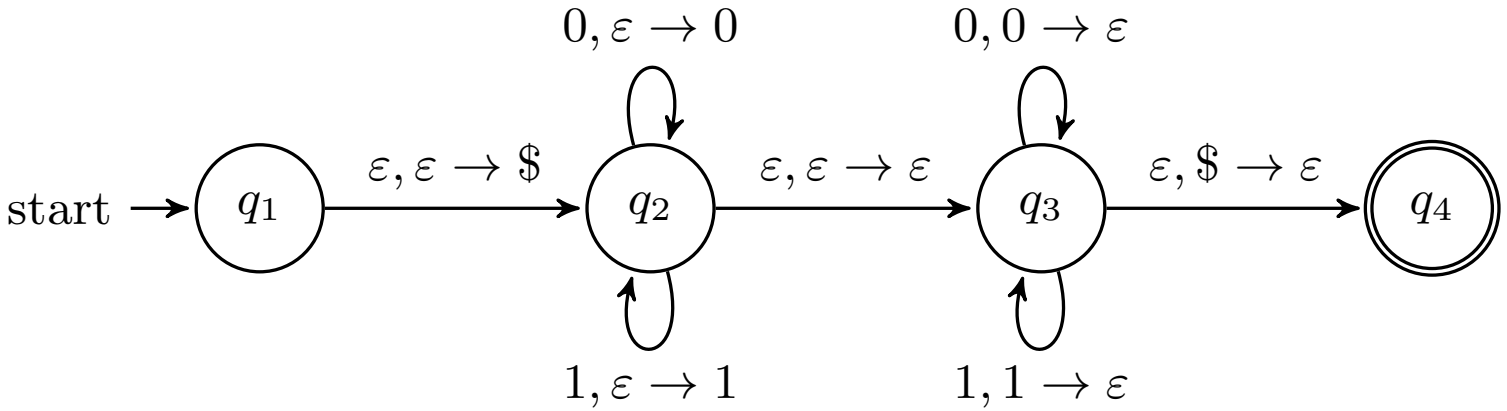
First, we slightly modify P to give it the following features:

1. It has a *single* accept state, q_f.
2. It *empties* its stack before accepting.
3. Each transition either pushes a symbol onto or pops one off the stack, but NOT both at the same time. (Note: This applies to ε-moves, i.e. if P has an ε-move, modify P such that some "*dummy*" symbol (not in Σ) is pushed onto the stack and is popped off the stack immediately after.)

Let the PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_f\})$.
We construct the CFG G as follows:
The variables of G are $\{A_{pq} \mid p, q \in Q\}$.
The axiom is A_{q_0,q_f}.
The rules of G are of three types:

1. For each $p, q, r, s \in Q, t \in \Gamma$, and $a, b \in \Sigma_\varepsilon$, if $\delta(p, a, \varepsilon)$ contains (r, t) and $\delta(s, b, t)$ contains (q, ε), put the rule $A_{pq} \to aA_{rs}b$ in G.
2. For each $p, q, r, \in Q$, put the rule $A_{pq} \to A_{pr}A_{rq}$ in G.
3. For each $p \in Q$, put the rule $A_{pp} \to \varepsilon$ in G.

An Example



First, we slightly modify P to give it the following features:

1. It has a *single* accept state, q_f.
2. It *empties* its stack before accepting.
3. Each transition either pushes a symbol onto or pops one off the stack, but NOT both at the same time.

The variables of G are $\{A_{pq} \mid p, q \in Q\}$.
The axiom is A_{q_0, q_f}.
The rules of G are of three types:

1. For each $p, q, r, s \in Q, t \in \Gamma$, and $a, b \in \Sigma_\varepsilon$, if $\delta(p, a, \varepsilon)$ contains (r, t) and $\delta(s, b, t)$ contains (q, ε), put the rule $A_{pq} \to aA_{rs}b$ in G.
2. For each $p, q, r, \in Q$, put the rule $A_{pq} \to A_{pr}A_{rq}$ in G.
3. For each $p \in Q$, put the rule $A_{pp} \to \varepsilon$ in G.

REG and CFL

We showed that PDAs recognize the class of context free languages (CFL).

REG and CFL

We showed that PDAs recognize the class of context free languages (CFL).

The class REG (regular languages) is recognized by finite automaton (FA), and every FA is a PDA without a stack.

REG and CFL

We showed that PDAs recognize the class of context free languages (CFL).

The class REG (regular languages) is recognized by finite automaton (FA), and every FA is a PDA without a stack.

Ergo, every $L \in REG$ is also in class CFL, i.e.

Theorem

Every regular language is a context free language.

Fin (wakas)

Thanks for the attention.

Questions?

Reading assignment(s)

[Sipser 2005] Chapter 2.2, or

[Hopcroft et al 2001] Chapter 6.3

References:

[Sipser 2005] M. Sipser. *Introduction to the Theory of Computation*: 2ed. PWS Publishing Company, 2005.

[Hopcroft, Ullman 1979] J. Hopcroft, J. Ullman. *Introduction to Automata Theory, Languages, and Computation*: Addison-Wesley, 1979.

[Hopcroft et al 2001] J. Hopcroft, R. Motwani, J. Ullman. *Introduction to Automata Theory, Languages, and Computation*: Addison-Wesley, 2001.