Harvard CS 121 and CSCI E-207
Lecture 10: Ambiguity, Pushdown Automata

Salil Vadhan

October 4, 2012

- **Reading:** Sipser, §2.2.

Another example of a CFG (with proof)

- $L = \{x \in \{a, b\}^* : x$ has the same # of a's and b's$\}$.

More examples of CFGs

- Arithmetic Expressions

G_1:

$$E \to x \mid y \mid E * E \mid E + E \mid (E)$$

G_2:

$$\begin{aligned} E &\to T \mid E + T \\ T &\to T * F \mid F \\ F &\to (E) \mid x \mid y \end{aligned}$$

Q: Which is “preferable”? Why?

Parse Trees

Derivations in a CFG can be represented by parse trees.

Examples:

Each parse tree corresponds to many derivations, but has unique leftmost derivation.

Parsing

Parsing: Given $x \in L(G)$, produce a parse tree for x. (Used to 'interpret' x. Compilers parse, rather than merely recognize, so they can assign semantics to expressions in the source language.)

Ambiguity: A grammar is ambiguous if some string has two parse trees.
Example:

Context-free Grammars and Automata

What is the fourth term in the analogy:

Regular Languages : Finite Automata

as

Context-free Languages : ???

Regular Grammars

Hint: There is a special kind of CFGs, the **regular grammars**, that generate exactly the regular languages.

A CFG is **(right-)regular** if any occurrence of a nonterminal on the RHS of a rule is as the rightmost symbol.

Turning a DFA into an equivalent Regular Grammar

- Variables are states.
- Transition $\delta(P, \sigma) = R$ $\quad P \xrightarrow{\sigma} R$

  becomes $P \to \sigma R$
- If P is accepting, add rule $P \to \varepsilon$

Regular Grammars (cont.)

Example of DFA$\Rightarrow$ Regular Grammars:

$\{x : x$ has an even # of a's and an even # of b's$\}$.

Other Direction: Omitted.

Context-free Grammars and Automata

What is the fourth term in the analogy:

Regular Languages : Finite Automata

as

Context-free Languages : ???

Sheila Greibach, AB Radcliffe ’60 *summa cum laude*

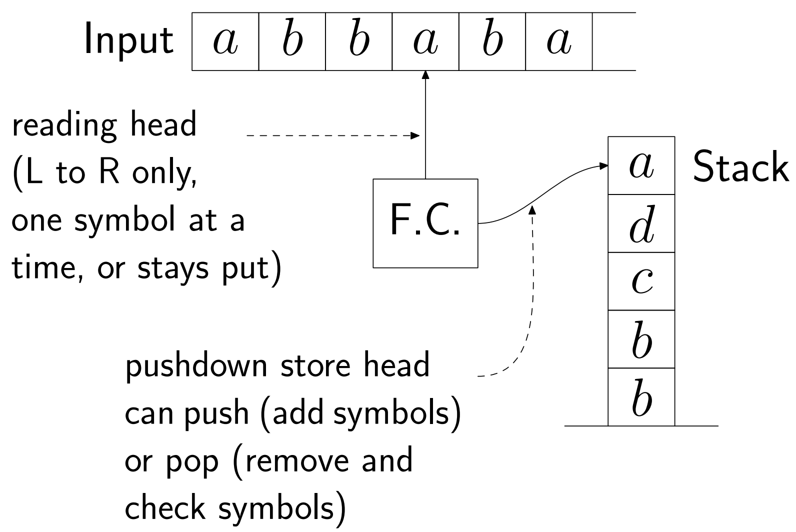
Inverses of Phrase Structure Generators

Harvard PhD Thesis, 1963

Pushdown Automata

= Finite automaton + "pushdown store"

- The pushdown store is a stack of symbols of unlimited size which the machine can read and alter only at the top.



Transitions of PDA are of form $(q, \sigma, \gamma) \mapsto (q', \gamma')$, which means:

If in state q with σ on the input tape and γ on top of the stack, replace γ by γ' on the stack and enter state q' while advancing the reading head over σ.

(Nondeterministic) PDA for "even palindromes"

$\{ww^R : w \in \{a,b\}^*\}$

$(q,a,\varepsilon) \mapsto (q,a)$	Push a's
$(q,b,\varepsilon) \mapsto (q,b)$	and b's
$(q,\varepsilon,\varepsilon) \mapsto (r,\varepsilon)$	switch to other state
$(r,a,a) \mapsto (r,\varepsilon)$	pop a's matching input
$(r,b,b) \mapsto (r,\varepsilon)$	pop b's matching input

So the precondition (q,σ,γ) means that

- the next $|\sigma|$ symbols (0 or 1) of the input are σ and
- the top $|\gamma|$ symbols (0 or 1) on the stack are γ

(Nondeterministic) PDA for "even palindromes"

$\{ww^R : w \in \{a,b\}^*\}$

$(q,a,\varepsilon) \mapsto (q,a)$	Push a's
$(q,b,\varepsilon) \mapsto (q,b)$	and b's
$(q,\varepsilon,\varepsilon) \mapsto (r,\varepsilon)$	switch to other state
$(r,a,a) \mapsto (r,\varepsilon)$	pop a's matching input
$(r,b,b) \mapsto (r,\varepsilon)$	pop b's matching input

Need to test whether stack empty: push $\$$ at beginning and check at end.

$$(q_0,\varepsilon,\varepsilon) \mapsto (q,\$)$$
$$(r,\varepsilon,\$) \mapsto (q_f,\varepsilon)$$

Language recognition with PDAs

A PDA <u>accepts</u> an input string

If there is a computation that starts

- in the start state
- with reading head at the beginning of string
- and the stack is empty

and ends

- in a final state
- with all the input consumed

A PDA computation becomes “blocked” (i.e. “dies”) if

- no transition matches *both* the input and stack

Formal Definition of a PDA

- $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

  $Q =$ states

  $\Sigma =$ input alphabet

  $\Gamma =$ stack alphabet

  $\delta =$ transition function

  $$Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to P(Q \times (\Gamma \cup \{\varepsilon\})).$$

  $q_0 =$ start state

  $F =$ final states

Computation by a PDA

- M accepts w if we can write $w = w_1 \cdots w_m$, where each $w_i \in \Sigma \cup \{\varepsilon\}$, and there is a sequence of states $r_0, \ldots, r_m$ and stack strings $s_0, \ldots, s_m \in \Gamma^*$ that satisfy

  1. $r_0 = q_0$ and $s_0 = \varepsilon$.
  2. For each i, $(r_{i+1}, \gamma') \in \delta(r_i, w_{i+1}, \gamma)$ where $s_i = \gamma t$ and $s_{i+1} = \gamma' t$ for some $\gamma, \gamma' \in \Gamma \cup \{\varepsilon\}$ and $t \in \Gamma^*$.
  3. $r_m \in F$.

- $L(M) = \{w \in \Sigma^* : M \text{ accepts } w\}$.

PDA for $\{w \in \{a,b\}^* : \#_a(w) = \#_b(w)\}$