Computer Science 513

Chomsky Hierarchy II: Context-Free Languages

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #22

Goals for Today

Goals for Today:

- Introduction to ***Context-Free Grammars and Languages***

Reference:

- Michael A. Harrison
  Introduction to Formal Language Theory
  Addison-Wesley, 1978

Context-Free Grammars

Consider a phrase-structure grammar

$$G = (V, \Sigma, \Pi, S)$$

as this is defined in Lecture #8.

Definition: *G* is a ***context-free grammar*** if each of the productions in Π has the form

$$A \to \mu$$

where $A \in V$ and $\mu \in (V \cup \Sigma)^\star$.

A language $L \subseteq \Sigma^\star$ is a ***context-free language*** if it is the language of a context-free grammar.

Context-Free Grammars

Example: Let $G = (V, \Sigma, \Pi, S)$ where

- $V = \{S\}$.
- $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.
- Π includes two productions, namely

$$S \to \mathtt{a}S\mathtt{b}$$

and

$$S \to \lambda$$

- S is the start variable.

Context-Free Grammars

- Since each of the productions in Π has the form "$A \to \mu$, for a variable $A \in V$ and a string $\mu \in (V \cup \Sigma)^\star$, G is a context-free grammar.
- It can be shown that the language of this grammar is

$$L(G) = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\} \subseteq \Sigma^\star.$$

Context-Free Grammars

Claim #1: Let $L \subseteq \Sigma^\star$ (for an alphabet Σ). If L is a regular language, then L is context-free.

Proof: Let $L \subseteq \Sigma^\star$ and suppose that L is a regular language. Then there exists a right-regular grammar

$$G = (V, \Sigma, \Pi, S)$$

such that $L = L(G)$. Comparing the definitions of a "right-regular grammar" and a "context-free grammar" one can see that the grammar G is also a context-free grammar — so that L is a context-free *language*, since $L = L(G)$. □

Context-Free Grammars

Claim #2: There exists a language $L \subseteq \Sigma^\star$ (for some alphabet Σ) such that L is context-free, but not regular.

Proof: Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. and let

$$L = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\} \subseteq \Sigma^\star.$$

Since L is the language of the context-free grammar, given in the above example, L is context-free. However — as noted in the previous lecture — the *Pumping Lemma for Regular Languages* can be used to prove that L is not a regular language — as is needed to establish the claim. □

Normal Forms

- Several ***normal forms*** for context-free languages — context-free grammars whose sets off productions have more limited forms — are known and used.
- The use of these simplify various tasks, including testing membership in context-free languages, and parsing.

Normal Forms

Definition: A context-free grammar $G = (V, \Sigma, \Pi, S)$ is in ***Chomsky Normal Form*** if every production in Π has one of the forms

$$A \to BC \qquad \text{or} \qquad A \to a$$

for a symbol $a \in \Sigma$ and for variables $A, B, C \in V$ — except that ***neither B nor C can be the start variable S.***

One more rule is allowed: If the empty string is in the language of the grammar then the rule

$$S \to \lambda$$

can (and, indeed, *must*) also be included.

Normal Forms

Claim #3: Let $G = (V, \Sigma, \Pi, S)$ be a context-free grammar. Then there exists a context-free grammar $\widehat{G} = (\widehat{V}, \Sigma, \widehat{\Pi}, \widehat{S})$ in Chomsky normal form such that $L(\widehat{G}) = L(G)$.

- A constructive proof of Claim #3 will be discussed in Lecture #24.
- It follows, by this claim, that every context-free language is the language of a context-free grammar that is in Chomsky normal form.

Pushdown Automata

Definition: A ***pushdown automaton (PDA)*** is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

where

- Q is a finite, nonempty set of ***states***.
- Σ is an ***alphabet***, used as a set of "input symbols" — such that $Q \cap \Sigma = \emptyset$. In the following, $\Sigma_\lambda = \Sigma \cup \{\lambda\}$.
- Γ is a finite, nonempty set of **pushdown symbols**.
- $q_0 \in Q$ is the ***start state***.
- Z_0 is the ***initial symbol*** on the "pushdown store" — which is a ***stack***.
- $F \subseteq Q$ is the set of ***accept states***.
- $\delta : Q \times \Sigma_\lambda \times \Gamma \to \mathcal{P}(Q \times \Gamma^\star)$ such that $\delta(q, \sigma, \gamma)$ is a ***finite*** set, for all $q \in Q$, $\sigma \in \Gamma_\lambda$ and for all $\gamma \in \Gamma$.

Pushdown Automata

Definition: An ***instantaneous description*** of a pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is an element

$$(q, \mu, \nu)$$

of the set $Q \times \Sigma^\star \times \Gamma^\star$. One can think of this instantaneous description as representing the following information:

- $q \in Q$ is the *current state* of M.
- $\mu \in \Sigma^\star$ is the part of the input string that remains to be processed (so that μ is a *suffix* of the string that M has been given as input).

Pushdown Automata

- $\nu \in \Gamma^\star$ is the current contents of the pushdown store — so that if $\nu = \lambda$ then the stack is empty. otherwise, if

$$\nu = \beta_1 \beta_2 \ldots \beta_k$$

for a positive integer k, β_1 is at the *bottom* of the stack, β_2 is immediately *above* it, and so on — so that β_k is at the *top* of the stack.

Pushdown Automata

Definition: Let (q_1, μ_1, ν_1) and (q_2, μ_2, ν_2) be instantaneous descriptions for a pushdown automaton M — so that $q_1, q_2 \in Q$, $\mu_1, \mu_2 \in \Sigma^\star$, and $\nu_1, \nu_2 \in \Gamma^\star$. Then $(q_1, \mu_1, \nu_1$ ***yields*** (q_2, μ_2, ν_2),

$$(q_1, \mu_1, \nu_1) \vdash (q_2, \mu_2, \nu_2),$$

if

- $\mu_1 = \sigma\mu_2$, for $\sigma \in \Sigma_\lambda = \Sigma \cup \{\lambda\}$,
- $\nu_1 = \widehat{\nu}\tau$, and $\nu_2 = \widehat{\nu}\rho$, for a pushdown symbol $\tau \in \Gamma$, and strings of pushdown symbols $\widehat{\nu}, \rho \in \Gamma^\star$, and
- $(q_2, \rho) \in \delta(q_1, \sigma, \tau)$.

Pushdown Automata

Thus, when a transition $(q_2, \rho) = \delta(q_1, \sigma, \tau)$ is applied,

- the symbol σ at the beginning of the current string is processed (unless $\sigma = \lambda$, in which case the current string is not changed,
- the pushdown symbol is "popped"' off the top of the stack, and
- the string ρ is "pushed" onto the stack.

Pushdown Automata

Definition: Let (q_1, μ_1, ν_1) and (q_2, μ_2, ν_2) be instantaneous descriptions for a pushdown automaton M — so that $q_1, q_2 \in Q$, $\mu_1, \mu_2 \in \Sigma^\star$, and $\nu_1, \nu_2 \in \Gamma^\star$.
Then (q_1, μ_1, ν_1) ***derives*** (q_2, μ_2, ν_2), if either

$$(q_1, \mu_1, \nu_1) = (q_2, \mu_2, \nu_2)$$

or there exists a positive integer k and a sequence of transitions $(\widehat{q}_i, \widehat{\mu}_i, \widehat{\nu}_i)$, for $0 \leq i \leq k$, such that

- $(q_1, \mu_1, \nu_1) = (\widehat{q}_0, \widehat{\mu}_0, \widehat{\nu}_0)$,
- $(\widehat{q}_i, \widehat{\mu}_i, \widehat{\nu}_i) \vdash (\widehat{q}_{i+1}, \widehat{\mu}_{i+1}, \widehat{\nu}_{i+1})$ for every integer i such that $0 \leq i \leq k - 1$, and
- $(\widehat{q}_k, \widehat{\mu}_k, \widehat{\nu}_k) = (q_2, \mu_2, \nu_2)$.

Pushdown Automata

There are at least *three* "languages of a pushdown automaton" that can be defined.

Definition: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a pushdown automaton.

- $T(M) \subseteq \Sigma^\star$ is the set of strings $\omega \in \Sigma^\star$ such that $(q_0, \omega, Z_0) \vdash^\star (r, \lambda, \rho)$ for some state $r \in F$ and some string of pushdown symbols $\rho \in \Gamma^\star$. This is the language ***accepted by M by final state***.
- $N(M) \subseteq \Sigma^\star$ is the set of strings $\omega \in \Sigma^\star$ such that $(q_0, \omega, Z_0) \vdash^\star (r, \lambda, \lambda)$ for some state $r \in Q$. This is the language ***accepted by M by empty store***.
- $L(M) \subseteq \Sigma^\star$ is the set of strings $\omega \in \Sigma^\star$ such that $(q_0, \omega, Z_0) \vdash^\star (r, \lambda, \lambda)$ for some state $r \in F$. This is the language ***accepted by M by both final state and empty store.***

Pushdown Automata

Claim #4: Let $L \subseteq \Sigma^\star$ (for an alphabet Σ). The following are equivalent:

- L is context-free.
- There exists a pushdown automaton M such that $L = T(M)$.
- There exists a pushdown automaton M such that $L = N(M)$.
- There exists a pushdown automaton M such that $L = L(M)$.

Pushdown Automata

- Different references sometimes define pushdown automata, and their languages, in somewhat different ways.
- With that noted, claims that are "essentially" the same as Claim #4 are stated in each of the following references:
  - Michael Sipser, *Introduction to the Theory of Computation* (Third Edition) — Section 2.2
  - John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, *Automata Theory, Languages, and Computation* (Third Edition) — Section 6.3
  - Michael A. Harrison, *Introduction to Formal Language Theory*, Section 5.5

  You should certainly be able to find a proof of this claim in other references, as well.

Languages That are *Not* Context-Free

Pumping Lemma for Context-Free Languages: Let $L \subseteq \Sigma^\star$. If L is a context-free language then there exists a positive integer p — called the ***pumping length*** for L — such that the following property is satisfied: For every string $s \in L$ such that $|s| \geq p$, there exist strings $u, v, w, x, y \in \Sigma^\star$ such that $s = uvwxy$ and the following properties are satisfied.

(i) $|vwx| \leq p$ — that is, this *middle* part of s is not very long.

(ii) $|vx| \geq 1$ — so that either $v \neq \lambda$ or $x \neq \lambda$ (or both).

(iii) $uv^i wx^i y \in L$ for every number $i \in \mathbb{N}$.

If there is time, a proof of this result will be sketched in the lecture presentation.

Languages That are *Not* Context-Free

- Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and let

$$L = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \in \mathbb{N}\}.$$

  The "Pumping Lemma for Context-Free Languages" can be used to prove that L is ***not*** context-free.
- If there is time then a proof that the above language is not context-free will be sketched during the lecture presentation.

Closure Properties

Claim #6: Let Σ be an alphabet and let $L_1, L_2 \subseteq \Sigma^\star$.

(a) If L_1 and L_2 are context-free then $L_1 \cup L_2$ is context-free.

(b) If L_1 and L_2 are context-free then $L_1 \circ L_2$ is context-free.

(c) If L_1 is context-free then $L_1^\star$ is context-free.

Claim #7: Let Σ be an alphabet such that $|\Sigma| \geq 2$.

(a) There exist languages $L_1, L_2 \subseteq \Sigma^\star$ such that L_1 and L_2 are both context-free, but $L_1 \cap L_2$ is ***not*** context-free.

(b) There exists a language $L \subseteq \Sigma^\star$ such that L is context-free, but the complement of L, $L^C = \{\omega \in \Sigma^\star \mid \omega \notin L\}$, is ***not*** context-free.

Proving these claims are included in the exercises for this lecture.

Decidable Problems and Computable Functions

A ***parse tree*** for a context-free grammar G is a rooted tree showing that a string is in the language of G.

- The root of the tree is labelled by the start variable.
- The children of an internal node with label A are labelled by elements of $V \cup \Sigma \cup \{\lambda\}$. Read in order from left to right, their labels form a string $\mu \in (V \cup \Sigma)^\star$ such that

$$A \to \mu$$

  is a production in the grammar.
- When read in order from left to right, the labels of the leaves form a string $\omega \in \Sigma^\star$ that is in the language of the grammar.

Decidable Problems and Computable Functions

A variety of problems, concerning context-free grammars and their languages are solvable (that is, languages are decidable — or functions are computable):

- ***Membership:*** Given a context-free grammar, $G = (V, \Sigma, \Pi, S)$ and a string $\omega \in \Sigma^\star$, decide whether $\omega \in L(G)$.
- ***Parsing:*** Given a context-free grammar, $G = (V, \Sigma, \Pi, S)$, and a string $\omega \in \Sigma^\star$ such that $\omega \in L(G)$, produce a ***parse tree*** for ω.

These problems will be consider in Lecture #24 and #25.

Decidable Problems and Computable Functions

- Given a context-free grammar $G = (V, \Sigma, \Pi, S)$, each of the following properties of the language of G can be determined:
  - It is possible to determine whether $L(G)$ is ***empty***.
  - It is possible to determine whether $L(G)$ is ***finite***.

  See Chapter 8 of *Introduction to Formal Language Theory* for details.

Ambiguity and Inherent Ambiguity

Consider a context-free grammar $G = (V, \Sigma, \Pi, S)$ such that

- $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$
- $V = \{S, T, U, A, B, C, D\}$
- S is the start variable
- Π includes the following rules.

$$S \to T \qquad A \to \lambda \qquad B \to \mathtt{a}B$$
$$S \to U \qquad C \to cC \qquad B \to \lambda$$
$$T \to AC \qquad C \to \lambda \qquad D \to \mathtt{b}D\mathtt{c}$$
$$A \to \mathtt{a}A\mathtt{b} \qquad U \to BD \qquad D \to \lambda$$

It is possible to show that the language of this context-free grammar is

$$\{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i, j, k \in \mathbb{N} \text{ and either } i = j \text{ or } j = k \text{ (or both)}\}.$$

Ambiguity and Inherent Ambiguity

One way to see that $\mathtt{abc} \in L(G)$ is to consider the following derivation.

$$
\begin{aligned}
S &\Rightarrow T && \text{(using the production } S \to T\text{)} \\
&\Rightarrow AC && \text{(using the production } T \to AC\text{)} \\
&\Rightarrow \mathtt{a}A\mathtt{b}C && \text{(using the production } A \to \mathtt{a}A\mathtt{b}\text{)} \\
&\Rightarrow \mathtt{ab}C && \text{(using the production } A \to \lambda\text{)} \\
&\Rightarrow \mathtt{abc}C && \text{(using the production } C \to \mathtt{c}C\text{)} \\
&\Rightarrow \mathtt{abc} && \text{(using the production } C \to \lambda\text{).}
\end{aligned}
$$

Ambiguity and Inherent Ambiguity

Another way to see that $\texttt{abc} \in L(G)$ is to consider the following — very different — derivation.

$$
\begin{aligned}
S &\Rightarrow U && \text{(using the production } S \to U\text{)} \\
&\Rightarrow BD && \text{(using the production } U \to BD\text{)} \\
&\Rightarrow \texttt{a}BD && \text{(using the production } B \to \texttt{a}B\text{)} \\
&\Rightarrow \texttt{a}D && \text{(using the production } B \to \lambda\text{)} \\
&\Rightarrow \texttt{ab}D\texttt{c} && \text{(using the production } D \to \texttt{b}D\texttt{c}\text{)} \\
&\Rightarrow \texttt{abc} && \text{(using the production } D \to \lambda\text{).}
\end{aligned}
$$

These derivations give two ***different parse trees*** for the string "abc".

Ambiguity and Inherent Ambiguity

Definition: A context-free grammar G is ***ambiguous*** if there exists a string $\omega \in L(G)$ that has two more parse trees (for G).

- Since the string "`abc`" has at least two parse trees, the grammar G, given above is ambiguous.
- Note that ambiguity is a property of *grammars* — not of their language: Sometime a context-free language, that is the language of an unambiguous context-free grammar is also the language of a different context-free grammar that is *not* ambiguous.
- Ambiguity is an undesirable property of context-free grammars for some applications — so heuristics to "disambiguate" context-free grammars are known and used.

Ambiguity and Inherent Ambiguity

Unfortunately, these heuristics are not always effective (and no others can be, either):

Definition: A context-free language is ***inherently ambiguous*** if *every* context-free grammar, whose language is L, is an ambiguous context-free grammar.

- Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. The language

$$L = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i, j, k \in \mathbb{N} \text{ and either } i = j \text{ or } j = k \text{ (or both)}\} \subseteq \Sigma^\star$$

  is inherently ambiguous.
- See Chapter 7 of *Introduction to Formal Language Theory* for a proof of this claim, and for more information about ambiguous grammars and inherently ambiguous languages.

Undecidable Problems

Finally, a variety of interesting problems, concerning context-free grammars and languages, have proved to be ***undecidable***. A few of these are given below.

- Decide whether a given context-free grammar is ***ambiguous***.
- Decide whether the *language* of a given context-free grammar is ***inherently ambiguous***.
- Decide whether the language of a given context-free grammar is a ***regular language***.
- Given a pair of context-free grammars G_1 and G_2, with the same alphabet Σ, decide whether $L(G_1) \cap L(G_2) = \emptyset$.

See Chapter 8 of *Introduction to Formal Language Theory* for proofs of these claims, and additional details.

Applications

- Context-free grammars have a significant application that you make use of on a regular basis — the ***compilation of computer programs***.
- You will therefore learn more about these — and make use of them — when taking a course like CPSC 411 (Compiler Construction)