Computer Science 513

Chomsky Hierarchy I: Regular Languages

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #21

Goals for Today

Goals for Today:

- Introduction to the ***Chomsky Hierarchy***
- Introduction to ***Regular Grammars and Languages***

Reference:

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

Chomsky Hierarchy

- The ***Chomsky Hierarchy*** is a containment hierarchy of formal languages — characterized by grammars — with applications in system software (notably, ***program compilation***) and linguistics.
- Along with the set of languages of ***phrase-structure grammars*** (discussed in Lecture #8) — that is, the set of *computably enumerable* languages — this includes the following.
  - the languages of ***regular grammars***
  - the languages of ***context-free grammars***
  - the language of ***context-sensitive grammars***
- Each of these will be defined, and discussed, in the remaining lectures.

Regular Grammars

Consider a phrase-structure grammar

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

as this is defined in Lecture #8.

Definition: G is a ***right-regular grammar*** if each of the productions in Π has one of the following forms.

(i) $A \to \sigma$, where $A \in V$ and $\sigma \in \Sigma$

(ii) $A \to \sigma B$ where $A, B \in V$ and $\sigma \in \Sigma$

(iii) $A \to \lambda$ where $A \in V$.

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

- Right-regular grammars are also called ***right-linear grammars***.

Regular Grammars

Example: Let $G = (V, \Sigma, \Pi, S_0)$ where $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, $V = \{S_0, S_1, S_2\}$, S_0 is the start variable, and Π includes the following productions:

$$\begin{array}{lll} \text{(i)}\ S_0 \to \mathtt{a}S_2 & \text{(iv)}\ S_2 \to \mathtt{a}S_1 & \text{(vi)}\ S_1 \to \mathtt{a}S_0 \\ \text{(ii)}\ S_0 \to \mathtt{b}S_0 & \text{(v)}\ S_2 \to \mathtt{b}S_2 & \text{(vii)}\ S_1 \to \mathtt{b}S_1 \\ \text{(iii)}\ S_0 \to \lambda & & \end{array}$$

- Since each of the productions has one of the forms given in the definition of a "right-regular grammar", *G is* a right-regular grammar.
- It can be shown that the *language* of this grammar is

$$L(G) = \{\omega \in \Sigma^\star \mid \text{the number of copies of "a" in } \omega \text{ is divisible by } 3\}.$$

Automata

Recall that a ***deterministic finite automaton (DFA)*** is (formally modelled as) a 5-tuple

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

where

- Q is a finite, nonempty set of ***states***.
- Σ is an ***alphabet***. Renaming states, as needed, we will require that $Q \cap \Sigma = \emptyset$.
- $\delta : Q \times \Sigma \to Q$ is a total function — called the ***transition function***.
- $q_0 \in Q$. This state is called the ***start state***.
- $F \subseteq Q$. This set is called the set of ***accept states***.

Automata

- The transition function, δ is use to define another total function, namely, an ***extended transition function*** $\delta^\star : Q \times \Sigma^\star \to Q$ such that, for all $q \in Q$ and for all $\omega \in \Sigma^\star$,

$$\delta^\star(q, \omega) = \begin{cases} q & \text{if } \omega = \lambda, \\ \delta(\delta^\star(q, \mu), \sigma) & \text{if } \omega = \mu \cdot \sigma \text{ for } \mu \in \Sigma^\star \\ & \qquad \text{and } \sigma \in \Sigma. \end{cases}$$

- Let $\omega \in \Sigma^\star$.
  - M ***accepts*** ω if $\delta^\star(q_0, \omega) \in F$.
  - M ***rejects*** ω if $\delta^\star(q_0, \omega) \notin F$.
- The ***language*** of M is the language

$$L(M) = \{\omega \in \Sigma^\star \mid M \text{ accepts } \omega\}.$$

Automata

- Deterministic finite automata, and their properties, were studied in the prerequisite course CPSC 351 (or CPSC 313).
- It was proved, in the prerequisite courses, that the following properties of a language $L \subseteq \Sigma^\star$ (for an alphabet Σ) are equivalent:
  - L is the language of a ***deterministic finite automaton***.
  - L is the language of a ***nondeterministic finite automaton.***
  - L is the language of a ***regular expression***.

These properties are useful when proving the following.

Automata

Claim #1: Let $L \subseteq \Sigma^\star$ (for an alphabet Σ). Then the following properties are equivalent:

- L is the language of a right-regular grammar G — so that L is a ***regular language***, as defined in these notes.
- L is the language of a deterministic finite automaton — so that L is a "regular language", as these were defined in CPSC 351 (or CPSC 313).

A proof of this claim is discussed in the lecture presentation.

Characterizations Using Different Kinds of Phrase-Structure Grammars

Once again, consider a phrase-structure grammar

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

Definition: G is a ***left-regular grammar*** if each of the productions in Π has one of the following forms.

(i) $A \to \sigma$, where $A \in V$ and $\sigma \in \Sigma$

(ii) $A \to B\sigma$ where $A, B \in V$ and $\sigma \in \Sigma$

(iii) $A \to \lambda$ where $A \in V$.

Characterizations Using Different Kinds of Phrase-Structure Grammars

Claim #2: Let $L \subseteq \Sigma^\star$ (for an alphabet Σ). Then the following properties are equivalent:

- L is the language of a right-regular grammar G — so that L is a regular language.
- L is the language of a left-regular grammar $\widehat{G}$.

A proof of this claim can be obtained by completing the exercises for this lecture.

Languages That are *Not* Regular

Pumping Lemma for Regular Languages: Let $L \subseteq \Sigma^\star$. If L is a regular 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 $x, y, z \in \Sigma^\star$ such that $s = xyz$ and the following properties are satisfied.

(i) $xy^i z \in L$ for every integer $i \geq 0$.

(ii) $|xy| \leq p$.

(iii) $|y| \geq 1$ (so that $y \neq \lambda$).

Languages That are *Not* Regular

- This result has (ideally) been presented in a prerequisite for this course and used to prove that languages are ***not*** regular.
- For example, let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ Then the Pumping Lemma for Regular Languages can be used to prove that the language

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

  is ***not*** a regular language.

Decidable Problems and Computable Functions

The proof, that every regular language is the language of a deterministic finite automaton (given in the lecture presentation) is ***constructive***.

- That is, it provides a ***process*** that can be followed to begin with a right-regular grammar $G = (V, \Sigma, \Pi, S)$ and produce a deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L(G)$.
- This can be used to obtain algorithms solving a variety of computational problems concerning right-regular grammars.
- A few computational problems, that can be solved using this approach, are listed on the following slides.

Decidable Problems and Computable Functions

- ***Membership:*** Given a right-regular grammar, $G = (V, \Sigma, \Pi, S)$ and a string $\omega \in \Sigma^\star$, decide whether $\omega \in L(G)$.
- ***Parsing:*** Given a right-regular grammar, $G = (V, \Sigma, \Pi, S)$, and a string $\omega \in \Sigma^\star$ such that $\omega \in L(G)$, produce a ***derivation*** of ω from the start variable, S, using productions in Π.
- ***Emptiness:*** Given a right-regular grammar $G = (V, \Sigma, \Pi, S)$, decide whether $L(G) = \emptyset$.

Decidable Problems and Computable Functions

- ***Everything?:*** Given a right regular grammar, $G = (V, \Sigma, \Pi, S)$, decide whether $L(G) = \Sigma^\star$.
- ***Containment:*** Given right-regular grammars $G_1 = (V_1, \Sigma, \Pi_1, S_1)$ and $G_2 = (V_2, \Sigma, \Pi_2, S_2)$, with the same alphabet Σ, decide whether $L(G_1) \subseteq L(G_2)$.
- ***Equivalence:*** Given right-regular grammars $G_1 = (V_1, \Sigma, \Pi_1, S_1)$ and $G_2 = (V_2, \Sigma, \Pi_2, S_2)$, with the same alphabet Σ, decide whether $L(G_1) = L(G_2)$.

Applications

- In applications, regular languages are generally given as the languages of ***regular expressions*** — but other representations of these (including finite automata) are needed to carry out computations.
- As the Wikipedia page for regular expressions states, "regular expressions are used in ***search engines***, in search and replace dialogues of ***word processors*** and ***text editors***, in text processing utilities such as `sed` and `AWK`, and in ***lexical analysis***,