CS330 Activities for Lecture 18

8.0 Languages and Grammars

- **A Language** is a set of words or sentences over some alphabet of symbols.
  - Natural Language – written, spoken, complicated rules of syntax; hard to write programs that understand them
  - Formal Language – specified by well-defined set of rules of syntax; useful in study of programming languages

- **A Grammar** is a set of rules used to
  - Generate the words/sentences of a language
  - Determine whether a word/sentence is in the language (= parsing; important for programming languages and compiler theory)
  - Grammars are concerned with syntax (formats), not semantics (meanings)
    - “The cow jumped over the moon” is semantically incorrect English
    - “The jumped cow the over moon” is syntactically incorrect English

**Example 1: Subset of English**

- We describe the sentences of a language using a grammar. This subset of English is defined using a list of rules that describe how a valid sentence can be produced. The names in *Capitalized Italics* are for syntactic categories.
  1. A Sentence is (made up of) a **Noun-Phrase** followed by a **Verb-Phrase**
  2. A **Noun-Phrase** is an **Article** followed by an **Adjective** followed by a **Noun**, or
  3. A **Noun-Phrase** is an **Adverb** followed by a **Noun**
  4. A **Verb-Phrase** is a **Verb** followed by an **Adverb**, or
  5. A **Verb-Phrase** is a **Verb**
  6. An **Article** is a, or
  7. An **Article** is the
  8. An **Adjective** is large, or
  9. An **Adjective** is hungry
  10. A **Noun** is rabbit, or
  11. A **Noun** is mathematician
  12. A **Verb** is eats, or
  13. A **Verb** is hops;
  14. An **Adverb** is quickly, or
  15. An **Adverb** is wildly

- From these rules we can form valid sentences using a series of replacements until no more rules can be used. Example:
  The sequence of replacements shows a **derivation** of the large rabbit hops quickly from Sentence:

<table>
<thead>
<tr>
<th>Noun-Phrase Verb-Phrase</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Article Adjective Noun Verb-Phrase</td>
<td>The start symbol</td>
</tr>
<tr>
<td>Article Adjective Noun Verb Adverb</td>
<td>Sequence of 2 nonterminals (syntactic categories)</td>
</tr>
<tr>
<td>the Adjective Noun Verb Adverb</td>
<td>4 nonterminals</td>
</tr>
<tr>
<td>the large noun verb Adverb</td>
<td>5 nonterminals</td>
</tr>
<tr>
<td>the large rabbit verb Adverb</td>
<td>One terminal followed by 4 nonterminals</td>
</tr>
<tr>
<td>the large rabbit hops Adverb</td>
<td>2 terminals followed by 3 nonterminals</td>
</tr>
<tr>
<td>the large rabbit hops quickly</td>
<td>3 terminals followed by 2 nonterminals</td>
</tr>
<tr>
<td>the large rabbit hops quickly</td>
<td>4 terminals followed by 1 nonterminals</td>
</tr>
<tr>
<td>the large rabbit hops quickly</td>
<td>5 terminals — nothing else to replace</td>
</tr>
</tbody>
</table>

---

* Some material from MATH230 Dr. Robert Ellis IIT-AMAT, zyBooks Disc. Math., and Disc. Math. and its Appl., Rosen (7th Ed.) is included under the academic/nonprofit Fair Use exception of US Copyright Law. This document must NOT be distributed beyond participants in IIT’s CS330.
• How to specify a Language
  • List all the words in a language — not practical in most cases
  • Give rules that a word must satisfy to be in the language — Language recognition
  • Specify a language using a set of rules — Language generation

• A Phrase-Structure Grammar $G$ consists of
  • Vocabulary $V$ — A set of symbols used to derive members of the language (Example: {a, b, ..., Noun, ...})
    • Define $V^*$ to be the set of all finite-length sequences of symbols from $V$
    • $V^*$ is pronounced “vee star” and the star is called the Kleene star
  • $V$ is partitioned into
    • Non-terminals $N$ — The symbols that can be replaced by other symbols (example: Noun)
    • Terminals $T$ — The symbols that can’t be replaced by other symbols (example: letter a)
  • Productions $P$ — Set of rules that specify when we can replace a string from $V^*$ with another string
    • Example: Sentence $\rightarrow$ Noun-Phrase Verb-Phrase
  • Start Symbol $S$ — The nonterminal that all derivations start with. (Example: Sentence, in our earlier example.)
    • Use the symbol $S$ if a start symbol isn’t explicitly given
  • A grammar $G$ generates a language $L(G)$, the set of all terminal sequences you can get using $G$.

Example 2 Language for a Grammar
• Given grammar $G = (V, T, P)$  
  • $N = V - T$, use start symbol $S$ by default]
    • $V = \{a, b, A, B, S\}$, $T = \{a, b\}$, $P = \{S \rightarrow ABa, A \rightarrow BB, B \rightarrow a b, AB \rightarrow b\}$
    • $L(G) = \{ba, abababa\}$
  • Once you get used to grammars, people often leave out $G$, $V$, and $T$ and just give the productions, with the understanding that the first production is for the start symbol, and some sort of convention for terminals and the names of nonterminals.
    • “What is the language generated by $S \rightarrow ABa$, $A \rightarrow BB$, $B \rightarrow a b$, $AB \rightarrow b$?”

Questions
1. Find the language of each grammar $G = (V, T, P)$
   1a. $V = \{a, b, A, S\}$, $T = \{a, b\}$, $P = \{S \rightarrow aA, S \rightarrow b, A \rightarrow a a\}$
   1b. $V = \{0, 1, S\}$, $T = \{0, 1\}$, $P = \{S \rightarrow 11S, S \rightarrow 0\}$
Example 3: Construct the grammar for the language

- (CS 440 and CS 532 discuss construction techniques for grammars)
- \( L(G) = \{0^n \, 1^n \mid n \in \mathbb{N}\} \) [each string has \( n \) zeros followed by \( n \) ones]
- \( V = \{0, 1, S\}, T = \{0, 1\}, P = \{S \rightarrow \lambda, S \rightarrow 0 \, S \, 1\}\)

Question

2. Construct a grammar for the language \( L(G) = \{0^m \, 1^n \mid m, n \in \mathbb{N}\} \)

Types of Phase-Structure Grammars

- Classified by types of production rules allowed (Noam Chomsky)
- The type of grammar is related to which model of computing machine can recognize the class of languages. Each type includes the ones under it.
  - Type 0 – no restrictions
  - Type 1: Production rules must have the form \( w_1 \rightarrow \lambda \) OR \( w_1 \rightarrow w_2 \), where length of \( w_1 \) ≤ length of \( w_2 \)
  - Type 2: \( A \rightarrow w_2 \) (i.e., \( w_1 \rightarrow w_2 \) where \( w_1 \) is some single non-terminal)
    - Note it’s okay to have same nonterminal go to different right-hand-sides
  - Type 3: \( A \rightarrow a \, B, A \rightarrow a, \) or \( S \rightarrow \lambda \)
    - I.e., \( w_1 \rightarrow w_2 \) where \( w_1 = A \) and \( (w_2 = a \, B \) or \( w_2 = a), \) or where \( w_1 = S \) and \( w_2 = \lambda \)

- Names of types
  - Type 1: Context-sensitive grammar, generates a context-sensitive language
  - Type 2: Context-free grammar, generates a context-free language
  - Type 3: Regular grammar, generates a regular language

Question

3. Review previous examples and classify the grammar type.

† The empty string \( \lambda \) is the zero-length sequence of symbols. Concatenation of a string with \( \lambda \) gives you back the same string. Example: \( a \, b \, \lambda = \lambda \, a \, b = a \lambda \, b = a \, b \). Concatenating 0 occurrences of a thing gives you \( \lambda \) (e.g., \( b^0 = \lambda \)).
Derivation Trees
- A word in a language generated by a context-free grammar can be represented graphically using an ordered rooted tree called a **derivation tree** or **parse tree**. (Remember diagrammed sentences from high school?)
  - Root is start symbol
  - Internal vertices are non-terminal symbols
  - Leaves are terminal symbols
- To show a usage of a production \( A \rightarrow w \), we take the vertex labeled \( A \) and draw children from it to each symbol in \( w \), in order, left-to-right.

Parsing
- To parse something is to determine if it is in a language generated by a grammar. (Typically context-free grammar, in CS.)
  - **Top-down Parsing:** Begin with the starting symbol, and successively apply productions
  - **Bottom-up Parsing:** Begin with the word (string of terminal symbols), successively apply reverse of productions
  - With both approaches you may hit “dead ends” and need to “backtrack”

Example 4: Parsing Example
- Say \( V = \{a, b, c, A, B, C, S\} \), \( T = \{a, b, c\} \), \( P = \{S \rightarrow AB, A \rightarrow Ca, B \rightarrow Ba, B \rightarrow Cb, B \rightarrow b, C \rightarrow cb, C \rightarrow b\} \)
  - Is cbab a word in the language generated by this grammar?
  - Top-down derivation: \( S \rightarrow AB \rightarrow Ca \rightarrow Bb \rightarrow cbab \)
  - Bottom-up derivation: \( cbab \leftarrow Cb \leftarrow A \leftarrow AB \leftarrow S \)

Question
4. Prefix notation arithmetic expression
4a. Give the grammar for the language of all strings representing a prefix notation arithmetic expression over numbers. Assume only the four binary operators +, −, *, and /, and assume all numbers can be represented by the terminal \( T \).

4b. Then verify the correctness of the following using your production rules (show which rules are used and in what order).
   * \( * \rightarrow 5 / 3 4 + 2 7 \)