Modeling Computation

Language and Grammars

- Natural Languages: English, French, German,…
- Programming Languages: Pascal, C, Java, …
- Grammar is used to generate sentences from basic words of the language
- Formal language: Generated by grammars.
- Why we study Formal languages ?
  - To model computation.. We will see the connection later.
  - Fundamental for compiler and study of programming languages
  - Natural language understanding, machine translation, human computer interaction.

What's the difference between these sentences:

Colorless green ideas sleep furiously.

Frankly walking pounded.

- Could a computer program distinguish a grammatical sentence from an ungrammatical one?

- Two answers: Yes and no.

- Yes: Linguists have shown that a remarkable amount of the structure of natural language can be captured with formal rules. Computer programs have been written that do a remarkably good job of recognizing valid sentences.

- No: But not all of it. Part of it seems tied to actually understanding the meaning, which is beyond us.

Syntax and Semantics

- Syntax: form of the sentence
- Semantics: the meaning of the sentence

The frog writes neatly

- Valid sentence grammatically (syntax)
- Noun phrase "the frog"
  - Made up of article "the" and noun "frog"
- Verb phrase "writes neatly"
  - Made up of the verb "writes" and adverb "neatly".

Swim quickly mathematics

- Is not a valid sentence

- Syntax for natural languages is extremely complicated
- Formal Language: specified by a well-defined set of rules of syntax (Grammar)
- The use of grammar helps to answer the following questions:
  - How can we determine whether a combination or words is a valid sentence
  - How can we generate the valid sentences of a formal language.

Example of a formal language:

1. **sentence**: **noun phrase** followed by **verb phrase**
2. **noun phrase**: **article** followed by **adjective** followed by **noun**
3. OR **articl**e followed by **noun**
4. **verb phrase**: **verb** followed by **adverb**
5. OR **verb**
6. article: *a*
7. OR article: *the*
8. adjective: *large*
9. OR *hungry*
10. noun: *rabbit*
11. OR *mathematician*
12. verb: *eats*
13. OR *hops*
14. adverb: *quickly*
15. OR *wildly*

- We can use these rules to generate valid sentences using a series of replacements of the bold terms:

- **sentence**
- **noun phrase verb phrase**
- **article adjective noun verb phrase**
- **article adjective noun verb adverb**
- *the* **adjective noun verb adverb**
- *the large* **noun verb adverb**
- *the large rabbit* **verb adverb**
- *the large rabbit hops* **adverb**
- the large rabbit hops quickly

- Other valid sentences are:
  - *A hungry mathematician eats widely*
  - *A large mathematician hops*
  - *The rabbit eats quickly*
- Invalid : *the quickly eats mathematician*

Definitions

- A “vocabulary” (alphabet) is a finite, nonempty set of elements called symbols
- A “sentence” (word) is a string of finite length of elements of V
- The “empty/null string” is denoted by lambda λ (zero length string)
- V*: the set of all sentences (words) defined over V
- A “language” is a subset of V*. (Kind of like a predicate on strings.)

- Vocabulary (alphabet) consists of terminals and nonterminals
- Terminals T: elements that can not be replaced by other symbols
- Nonterminals N: elements that can be replaced by other symbols.

- In our previous example:
- V= {**sentence, noun phrase, verb phrase, article, adjective, noun, verb, adverb,** *a, the, large, hungry, rabbit, mathematician, eats, hops, quickly, wildly*}
- T = {*a, the, large, hungry, rabbit, mathematician, eats, hops, quickly, wildly*}
- N= {**sentence, noun phrase, verb phrase, article, adjective, noun, verb, adverb** }

Phrase-structure grammar

A phrase-structured grammar

G = (V, T, S, P) consists of:

V: vocabulary

T : a subset of V called terminal symbols

N=V-T: non-terminals

S in V: start symbol

P: productions (set of pairs of strings)

Every production in P must contain at least one nonterminal on its left side.

Example:

- G = (V,T,S,P)
- V = { a, b, A, B, S }
- T = { a, b }
- S is the start symbol
- P = { $S \rightarrow ABa$, $A \rightarrow BB$, $B \rightarrow ab$, $AB \rightarrow b$ }

What words can G generate?

Let $w_0 = l\ z_0\ r$

Let $w_1 = l\ z_1\ r$

- if $z_0 \rightarrow z_1$ is a production of G
  then $w_0 \Rightarrow w_1$ (w_1 is "directly derivable" from w_0)
- $w_0, w_1, ..., w_n$ are strings over V s.t.

$w_0 \Rightarrow w_1 \Rightarrow w_2 , ..., w_{\{n-1\}} \Rightarrow w_n$

then we say that w_n is "derivable" from w_0 , we write $w_0 \Rightarrow^* w_n$,

-
- The sequence of steps used is a "derivation".

Ex: Aaba is directly derivable from ABa

abababa is derivable from ABa

- "Language generated" from a grammar G, denoted by L(G):

$L(G) = \{ w \text{ in } T^* \mid S \Rightarrow^* w \}$