1 Languages

Recall that $\Sigma$ denotes a finite alphabet and $\Sigma^*$ the set of all strings of finite length constructable from this alphabet. A language is a subset of $\Sigma^*$.

Example 1 Let $\Sigma = \{C, V\}$. $C$ and $V$ are meant to invoke consonants and vowels, respectively. Some languages only allow syllables of exactly one consonant followed by exactly one vowel. Assuming that words are sequences of syllables, the ‘language’ of possible words which describes this phenomenon is

$$L = \{CV, CVC, CVCVC, \ldots\}$$

Example 2 Let $\Sigma = \{\text{John}, \text{laughed}, \text{and}\}$. 

$$L = \{\text{John laughed}, \text{John laughed and laughed}, \text{John laughed and laughed}, \ldots\}$$

Exercise 1 What are some other strings that do not belong to the two languages mentioned above?

The problem of how we decide whether strings belong to a language or not is called the membership problem.

Many linguists, including Chomsky, criticize defining languages as a subset of $\Sigma^*$. They argue that this perspective only cares whether strings belong to the language or not, which barely touches questions of linguistic interest. More important questions include:

1. What is the structure of a sentence?
2. How do (structured) sentences relate to their meaning(s)?

I’m not convinced by these arguments. We can incorporate structure into our trees by add to our alphabet the open and closed parenthesis. We can now ask whether $(\text{John (laughed and laughed)})$ belongs to English and whether $(\text{John laughed and) laughed})$ does.

How can we address meaning? Add to our alphabet the symbols we will need for the language we will use to describe what sentences mean. Add one more symbol to indicate the break between the sentence and its meaning. We can now ask whether $\text{John laughed:LAUGHED(JOHN)}$ and $\text{John laughed:CRIED(JOHN)}$ belongs to our language.

More generally, once we recognize that tree structures, and/or tree structures combined with semantic meaning, and/or entire syntactic derivations can be encoded as strings, we recognize how flexible and powerful the membership problem is.

There are other problems. For example, we may ask: given a grammar, and a string with all structural symbols (like parentheses) erased, what are the possible structures that this sentence can have? This is the parsing problem.

Exercise 2 We observed before that $\Sigma^*$ is countably enumerable. Is the set of all logically possible languages countably enumerable?
2 Grammars

Essentially, grammars are finite objects that describe some language. More precisely, if a language has a grammar, then we ought to be able to use the grammar to solve the membership problem.

There are many ways grammars can be defined. We will now define rewrite grammars. It will turn out that this choice of grammar formalism doesn’t matter so much because it is Turing-complete—which means that there are many other general computing systems that are equivalent to it. A rewrite grammar consists of the following collection of items:

• A nonempty finite alphabet of symbols $\Sigma$. These symbols are also called the terminal symbols, and we usually write them with lowercase letters like $a, b, c, \ldots$

• A nonempty finite set of non-terminal symbols, which denote $\Delta$. These symbols are also called category symbols, and we usually write them with uppercase letters like $A, B, C, \ldots$

• A start category $S$, which belongs to $\Delta$.

• A finite set of production rules $\mathcal{R}$. A production rule has the form $\alpha \rightarrow \beta$

where $\alpha, \beta$ belongs to $(\Sigma \cup \Delta)^*$. In other words, $\alpha$ and $\beta$ are strings of non-terminal and terminal symbols. Note $\beta$ may be the empty string but $\alpha$ must include at least one symbol.

Before discussing how these grammars generate languages, let’s just see some example grammars.

Example 3 Consider the following grammar $G_1$:

• $\Sigma = \{\text{john, laughed, and}\}$;

• $\Delta = \{S, VP1, VP2\}$; and

• $\mathcal{R} = \left\{ \begin{array}{l}
S \rightarrow \text{john } VP \\
VP1 \rightarrow \text{laughed } VP2 \\
VP2 \rightarrow \text{and } laughed \ VP2 \\
VP2 \rightarrow \lambda
\end{array} \right\}$

Example 4 Consider the following grammar $G_2$:

• $\Sigma = \{a, b\}$;

• $\Delta = \{S\}$; and
The production rules are
\[
\mathcal{R} = \left\{ \begin{array}{l}
S \rightarrow aSb \\
S \rightarrow \lambda
\end{array} \right\}
\]

Now how does a grammar generate a language? Assume we have some rewrite grammar \( G = (\Sigma, \Delta, S, R) \) with a production rule \( r \) that rewrites \( \alpha \) as \( \beta \). Then for all \( \gamma_1, \gamma_2 \) belonging to \((\Sigma \cup \Delta)^*\), the string of terminals and nonterminals \( \gamma_1\alpha\gamma_2 \) can be rewritten as \( \gamma_1\beta\gamma_2 \). We use \( \Rightarrow \) to indicate this.

\[
\gamma_1\alpha\gamma_2 \Rightarrow \gamma_1\beta\gamma_2
\]

The rules can apply as often as desired and in any order. For all \( \alpha, \beta \) belonging to \((\Sigma \cup \Delta)^*\), if the application of zero or more rules permits \( \alpha \) to be rewritten as \( \beta \), we write

\[
\gamma_1\alpha\gamma_2 \Rightarrow^* \gamma_1\beta\gamma_2
\]

Then the language generated by the grammar \( G \) is the set of all strings \( w \) belonging to \( \Sigma^* \) such that there is some way of applying the production rules such that \( S \Rightarrow^* w \).

Exercise 3 How does \( G_1 \) generate John laughed and laughed and laughed?

Exercise 4 What language does \( G_2 \) generate?

Exercise 5 Are all logically possible rewrite rule grammars countably infinite?

Exercise 6 Compare answers to exercise 5 with the answer to exercise 2. What does this mean?

The Church-Turing thesis states that anything that is computable is computable with a Turing machine or any equivalent programming system. Rewrite rule grammars are Turing-equivalent. So most things are not computable!

The central idea in cognitive science (and linguistics) is that the brain makes computations when processing information. What kinds of computations must be made to learn, to produce, and to understand language? What is the computational nature of natural language?

3 Summary of the lesson

1. The natural numbers are countably infinite. The powerset of natural numbers is not (it has strictly more elements).

2. Likewise, the set of logically possible language grammars is countably infinite, but the set of logically possible languages is not.

3. Languages generable by some grammar are called recursively enumerable. They are countably infinite, and therefore a small fraction of the logically possible languages.