Chapter 11

Context Free Grammars

11.1 Why, what's wrong with FSAs?

We have spent a long time talking about all the things that FSAs (and FSTs) can do, but there are important things that they can't do. For example, they cannot generate the language

$$L = \{a^n b^n \mid n \geq 0\} \tag{11.1}$$

Question 13. Intuitively, why can't FSAs do this?

Question 14. How would you prove this rigorously? The standard argument uses the *pumping lemma*. The argument starts like this: suppose there is a deterministic FSA with m states that generates L. Among the strings it accepts, it must accept the string $a^m b^m$. Now, when the automaton reaches the exact midpoint of the string, it has gone through $(m+1)$ states. But there are only m distinct states possible. So it must have gone through the same state twice, that is, it must have gone through a cycle. Let $k > 0$ be the number of a's that it read while going round the cycle. Can you think of another string *not* in L that the automaton must also accept, leading to a contradiction?

Linguistically, the analogous property that FSAs are missing is the ability to do *center-embedding*. English allows sentences like:

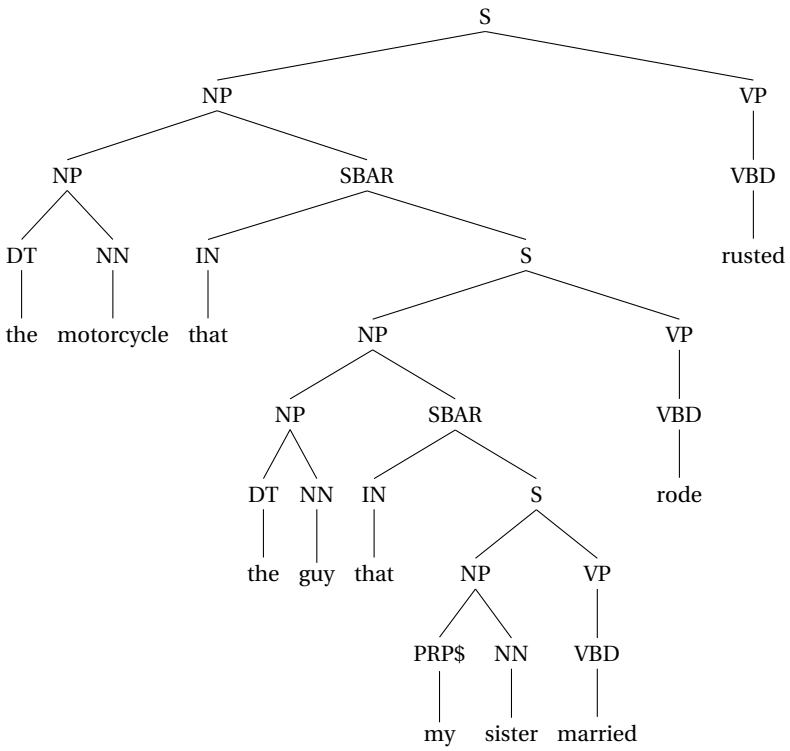
(11.2) The motorcyle rusted.

(11.3) The motorcycle that the guy rode rusted.

(11.4) The motorcycle that the guy that my sister married rode rusted.

At minimum, we need the number of noun phrases to equal the number of verbs, which we have already seen FSAs aren't able to do. Actually, we want to be able to create a structure like

(11.5)



which can tell us which noun corresponds to which verb. Later, we will see how this structure is also useful for translating into another language like Japanese or Hindi.

Question 15. The above argument holds only if you believe that *unbounded* center embedding is possible. In fact, center-embedding examples degrade rather quickly as more levels are added:

(11.6) The motorcycle that the guy that the sister that my mom spoiled married rode rusted.

If center embedding is bounded, then how would you write an FSA to model it? What would still be unsatisfactory about such an account?

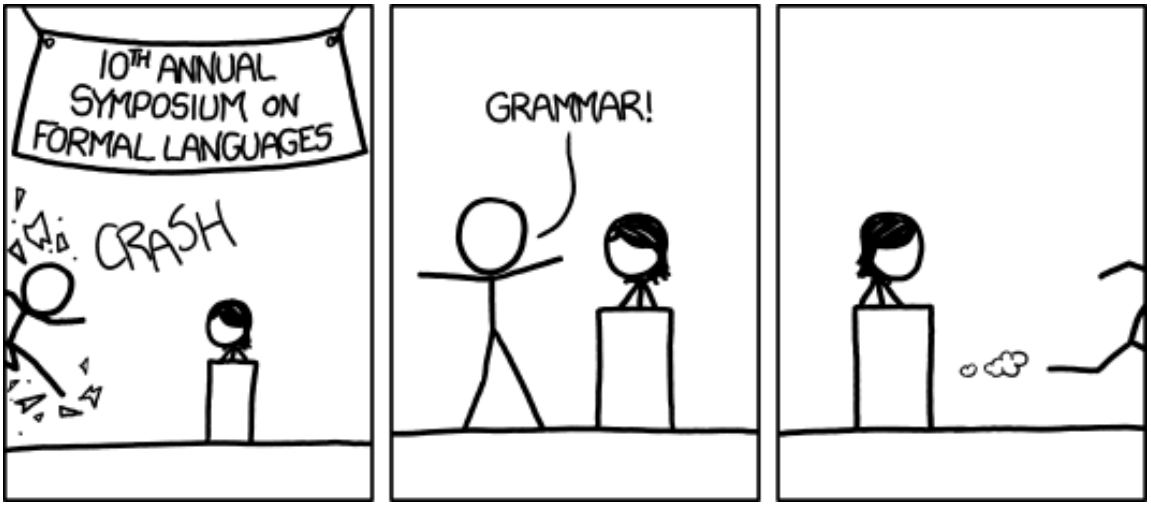


Figure 11.1: Formal Languages. [audience looks around] 'What just happened?' 'There must be some context we're missing.'

11.2 Context free grammars

Our solution is to use *context free grammars (CFGs)*. CFGs are also widely used in compilers, where they are known as *Backus-Naur Form*. We begin with two examples of CFGs. The first one generates the non-finite-state language $\{a^n b^n \mid n \geq 0\}$:

$$\begin{aligned} S &\rightarrow aSb \\ S &\rightarrow \epsilon \end{aligned} \tag{11.7}$$

Here, S is called a *nonterminal symbol* and can be rewritten using one of the above rules, whereas a and b are called *terminal symbols* and cannot be rewritten. This is how the grammar works: start with a single S, then repeatedly choose the leftmost nonterminal and rewrite it using one of the rules until there are no more nonterminals. For example:

$$\begin{aligned} S &\Rightarrow aSb \\ &\Rightarrow aaSbb \\ &\Rightarrow aaaSbbb \\ &\Rightarrow aaabbb \end{aligned} \tag{11.8}$$

Our next example generates sentences (11.2), (11.3), and (11.4).

$$
\begin{aligned}
\text{S} &\rightarrow \text{NP VP} \\
\text{NP} &\rightarrow \text{DT NN} \\
\text{NP} &\rightarrow \text{PRP\$ NN} \\
\text{NP} &\rightarrow \text{NP SBAR} \\
\text{VP} &\rightarrow \text{VBD} \\
\text{SBAR} &\rightarrow \text{IN S} \\
\text{DT} &\rightarrow \text{the} \\
\text{PRP\$} &\rightarrow \text{my} \\
\text{NN} &\rightarrow \text{motorcycle} \mid \text{guy} \mid \text{sister} \\
\text{VBD} &\rightarrow \text{married} \mid \text{rode} \mid \text{rusted} \\
\text{IN} &\rightarrow \text{that}
\end{aligned}
\tag{11.9}
$$

Here, the uppercase symbols are nonterminal symbols, and the English words are terminal symbols. Also, we have used some shorthand: $A \rightarrow \beta_1 \mid \beta_2$ stands for two rules, $A \rightarrow \beta_1$ and $A \rightarrow \beta_2$.

Question 16. How would you use the above grammar to derive sentence (11.3)?

Here's a more formal definition of CFGs.

Definition 3. A *context-free grammar* is a tuple $G = (N, \Sigma, R, S)$, where

- N is a set of *nonterminal symbols*
- Σ is a set of *terminal symbols*
- R is a set of *rules* or *productions* of the form $A \rightarrow \beta$, where $A \in N$ and $\beta \in (N \cup \Sigma)^*$
- $S \in N$ is a distinguished *start symbol*

If $A \in N$ and $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$, we write $\alpha A \gamma \Rightarrow_G \alpha\beta\gamma$ iff $(A \rightarrow \beta) \in R$, and we write $\Rightarrow_G^*$ for the reflexive, transitive closure of $\Rightarrow_G$. Then the language generated by G is $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.

11.3 Structure and ambiguity

As has already been alluded to, CFGs are interesting not only because they can generate more string languages than FSAs can, but because they build trees, known as *syntactic analyses*, *phrase-structure*

trees, or *parse trees*. Whenever we use a rule $A \to \beta$ to rewrite a nonterminal A, we don't erase A and replace it with β; instead, we make the symbols of β the children of A.

Question 17. What would the tree for sentence (11.3) be?

One of the main purposes of these trees is that every subtree of the parse tree is supposed to have a *semantics* or meaning, so that the tree shows how to interpret the sentence. As a result, it is possible that a single string can have more than one structure, and therefore more than one meaning. This is called *ambiguity*. To illustrate it, we need a new example.

$$
\begin{aligned}
\text{S} &\to \text{NP VP} \\
\text{NP} &\to \text{DT NN} \\
\text{NP} &\to \text{NN} \\
\text{NP} &\to \text{NN NNS} \\
\text{VP} &\to \text{VBP NP} \\
\text{VP} &\to \text{VBP} \\
\text{VP} &\to \text{VP PP} \\
\text{PP} &\to \text{IN NP} \\
\text{DT} &\to \text{a} \mid \text{an} \\
\text{NN} &\to \text{time} \mid \text{fruit} \mid \text{arrow} \mid \text{banana} \\
\text{NNS} &\to \text{flies} \\
\text{VBP} &\to \text{flies} \mid \text{like} \\
\text{IN} &\to \text{like}
\end{aligned}
\tag{11.10}
$$

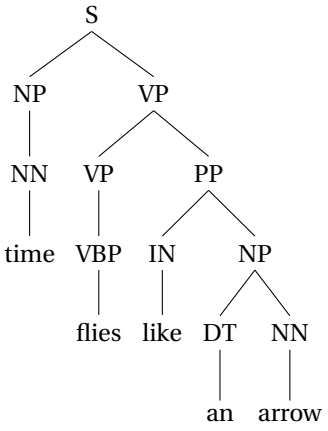
This grammar generates (among others) the following two strings:

(11.11) Time flies like an arrow.

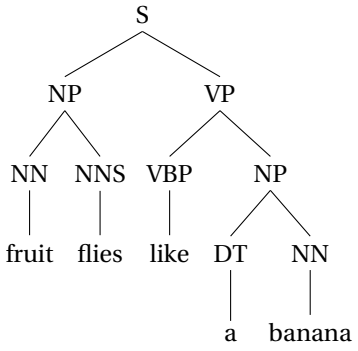
(11.12) Fruit flies like a banana.

Their "natural" structures are:

(11.13)

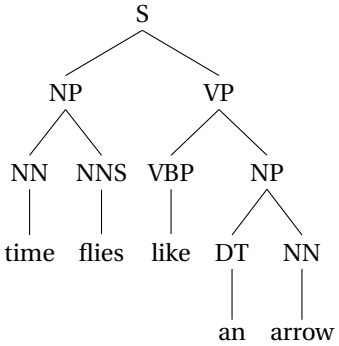


(11.14)

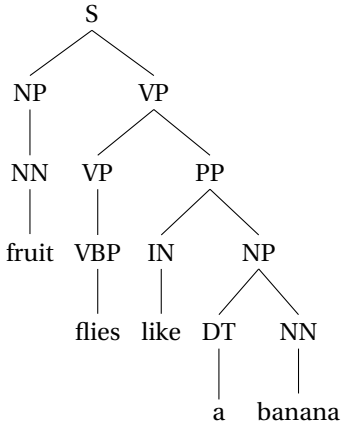


But the grammar also allows other structures, which would lead to other meanings:

(11.15)



(11.16)



Interpretation (11.15) says that a certain kind of fly, the time fly, is fond of arrows. Interpretation (11.16) says that fruits generally fly in the same way that bananas fly.

Question 18. The English word *buffalo* has two meanings: it can be a noun (the name of several species of oxen) or a verb (to overpower, overawe, or constrain by superior force or influence; to outwit, perplex). Also, the plural of the noun *buffalo* is *buffalo*. Therefore, the following strings are all grammatical:

(11.17) Buffalo! (Overpower!)

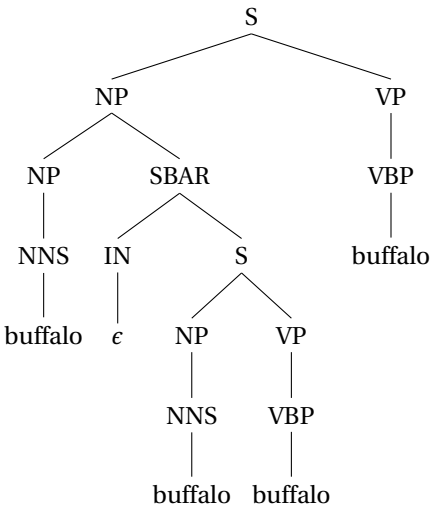
(11.18) Buffalo buffalo. (Oxen overpower.)

(11.19) Buffalo buffalo buffalo. (Oxen overpower oxen.)

(11.20) Buffalo buffalo buffalo buffalo. (Oxen that overpower oxen, overpower.)

In fact, the entire set $\{\text{buffalo}^n \mid n \geq 1\}$ is a subset of English. Can you write a CFG that generates it according to English grammar? Hint: here is a tree for the fourth example:

(11.21)



How many structures can you find for the following sentence:

(11.22) Buffalo buffalo buffalo buffalo buffalo.

11.4 Weighted context free grammars

Weighted CFGs are a straightforward extension of CFGs. Recall that FSTs map an input string to a set of possible output strings, whereas weighted FSTs give us a *distribution* over possible output strings. In the same way, weighted CFGs help us deal with ambiguity (a single string having multiple structures) by giving us a *distribution* over possible structures.

In a weighted CFG, every production has a weight attached to it, which we write as

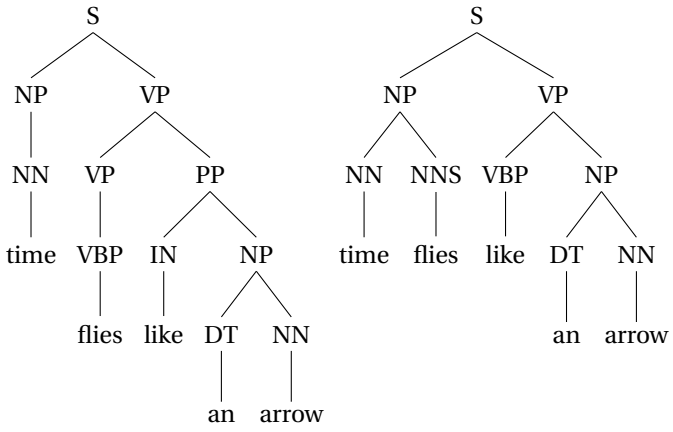
$$A \xrightarrow{p} \beta$$

The weight of a derivation is the product of the weights of the rules used in the derivation (if a rule is used k times, we multiply its weight in k times).

Thus we can take our grammar from last time and add weights:

$$
\begin{array}{ll}
 & \mathrm{DT} \xrightarrow{0.5} \text{a} \\
\mathrm{S} \xrightarrow{1} \mathrm{NP\ VP} & \mathrm{DT} \xrightarrow{0.5} \text{an} \\
\mathrm{NP} \xrightarrow{0.5} \mathrm{DT\ NN} & \mathrm{NN} \xrightarrow{0.25} \text{time} \\
\mathrm{NP} \xrightarrow{0.4} \mathrm{NN} & \mathrm{NN} \xrightarrow{0.25} \text{fruit} \\
\mathrm{NP} \xrightarrow{0.1} \mathrm{NN\ NNS} & \mathrm{NN} \xrightarrow{0.25} \text{arrow} \\
\mathrm{VP} \xrightarrow{0.6} \mathrm{VBP\ NP} & \mathrm{NN} \xrightarrow{0.25} \text{banana} \\
\mathrm{VP} \xrightarrow{0.3} \mathrm{VBP} & \mathrm{NNS} \xrightarrow{1} \text{flies} \\
\mathrm{VP} \xrightarrow{0.1} \mathrm{VP\ PP} & \mathrm{VBP} \xrightarrow{0.5} \text{flies} \\
\mathrm{PP} \xrightarrow{1} \mathrm{IN\ NP} & \mathrm{VBP} \xrightarrow{0.5} \text{like} \\
 & \mathrm{IN} \xrightarrow{1} \text{like}
\end{array}
\tag{11.23}
$$

Question What would the weight of these two derivations be?



A *probabilistic CFG* or *PCFG* is one in which the probabilities of all rules with a given left-hand side sum to one (Booth and Thompson, 1973). A PCFG is called *consistent* if the probabilities of all derivations sum to one.

Aren't all PCFGs consistent? Actually, no:

$$\mathrm{S} \xrightarrow{0.9} \mathrm{SS} \tag{11.24}$$

$$\mathrm{S} \xrightarrow{0.1} \mathrm{a} \tag{11.25}$$

Let P_n be the total weight of trees of height $\leq n$. Thus

$$P_1 = 0.1 \tag{11.26}$$

$$P_{n+1} = 0.9P_n^2 + 0.1 \tag{11.27}$$

The second equation is because a tree of height $\leq n+1$ can either be a tree of height 1, formed using rule (11.25), or a tree formed using rule (11.24) and two trees of height $\leq n$. The limit $P = \lim_{n\to\infty} P_n$ must be a fixed point of the second equation. There are two fixed points: $P = 0.9P^2 + 0.1 \Rightarrow P = \frac{1}{9}$ or 1. But note that $P_1 < \frac{1}{9}$, and $P_i < \frac{1}{9} \Rightarrow P_{i+1} < \frac{1}{9}$. Since the sequence is always less than $\frac{1}{9}$, it cannot converge to 1. Therefore, $P = \frac{1}{9}$!

Question What happened to the other $\frac{8}{9}$?

Bibliography

Bar-Hillel, Y., M. Perles, and E. Shamir (1961). "On formal properties of simple phrase structure grammars". In: *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 14.2, pp. 143–172.

Booth, Taylor L. and Richard A. Thompson (1973). "Applying Probability Measures to Abstract Languages". In: *IEEE Trans. Computers* C-22.5, pp. 442–450.