Review of context-free grammars

LING83600

1 Introduction

Context-free grammars (or CFGs) are a formalization of what linguists call "phrase structure grammars". The formalism is originally due to Chomsky (1956) though it has been independently discovered several times. Despite their limitations, CFG grammars of human languages are widely used as a model of syntactic structure in natural language processing and understanding tasks, and virtually all modern programming languages are described with, and parsed as, a CFG.[1]

Bibliographic note

This handout covers the formal definition of context-free grammars. Algorithms for parsing CFGs are covered in later courses. The definitions here are loosely based on Jurafsky and Martin in preparation, chap. 12, who in turn draw from Hopcroft and Ullman 1979.

2 Definitions

2.1 Context-free grammars

A context-free grammar G is a four-tuple N, Σ, R, S such that:

- N is a set of *non-terminal* symbols, corresponding to phrase markers in a syntactic tree.
- Σ is a set of *terminal* symbols, corresponding to words (i.e., X_0s) in a syntactic tree.
- R is a set of *production rules*. These rules are of the form $A \rightarrow \beta$ where $A \in N$ and $\beta \in (\Sigma \cup N)^*$. Thus A is a phrase label and β is a sequence of zero or more terminals and/or non-terminals.
- $S \in N$ is a designated start symbol (i.e., the highest projection in a sentence).

For simplicity, we assume N and Σ are disjoint. As is standard, we use Roman uppercase characters to represent non-terminals and Greek lowercase characters to represent terminals.

[1] For example, the following is the full grammar specification for Python 3: `https://docs.python.org/3/reference/grammar.html`. This file is actually used to generate a parser for the CPython interpreter.

2.2 Derivation

Direct derivation describes the relationship between the input to a single grammar rule in R and the resulting output. If there is a rule $A \rightarrow \beta \in R$, and α, γ are strings in $(\Sigma \cup N)^*$, then

$$\alpha A \gamma \Rightarrow \alpha \beta \gamma$$

i.e., $\alpha A \gamma$ directly derives $\alpha \beta \gamma$. *Derivation* is a generalization of direct derivation which allows us to iteratively apply rules to strings. Given strings $\alpha_1, \alpha_2, \alpha_m \in (\Sigma \cup N)^*$ such that $\alpha_1 \Rightarrow \alpha_2$, and $\alpha_2 \Rightarrow \alpha_3$, ..., $\alpha_{m-1} \Rightarrow \alpha_m$, then

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

i.e., α_1 *derives* α_m (and α_1 also derives α_2, α_3, etc.).

2.3 Context-free language

The language L_G generated by some grammar G is the (possibly infinite) set of strings of terminal symbols that can be derived by G starting from the start symbol S.

3 Chomsky-normal form

Syntacticians have long had a preference for *binary branching* syntactic structures, meaning that each non-terminal node has at most two children. This assumption greatly simplifies parsing algorithms as well. One way this is enforced by converting grammars or treebanks to a format known as *Chomsky normal form* (CNF; Chomsky 1963). In Chomsky normal form, the elements of R, the set of production rules, are constrained to have one of two forms:

- $A \rightarrow B\ C$ where $A, B, C \in N$.
- $A \rightarrow \beta$ where $A \in N$ and $\beta \in \Sigma$.

In other words, the right-hand side of every rule either consists of two non-terminals or one terminal. There exists for every CFG grammar a *weakly equivalent* CNF grammar, meaning that there exists a CNF which generates the same language (though it does not necessarily assign exactly the same phrase structure). For instance, given the rule $A \rightarrow B\ C\ D$, we can convert this to two CNF rules, namely $A \rightarrow B\ X$ and $X \rightarrow C\ D$.

References

Chomsky, Noam. 1956. Three models for the description of language. *IEEE Transactions on Information Theory* 3:113–124.

Chomsky, Noam. 1963. Formal properties of grammars. In *Handbook of Mathematical Psychology*, ed. R. Duncan Luce, Robert R. Bush, and Eugene Galanter, 323–418. John Wiley & Sons.

Hopcroft, John E., and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley.

Jurafsky, Dan, and James H. Martin. in preparation. *Speech and Language Processing*. 3rd edition.