22.5 DEFINITE CLAUSE GRAMMAR (DCG)

There are two problems with BNF. First, we are interested in using language for communication, so we need some way of associating a meaning with each string, and BNF only talks about strings, not meanings. Second, we will want to describe grammars that are context-sensitive, whereas BNF is strictly context-free. In this section, we introduce a formalism that can handle both of these problems.

The idea is to use the full power of first-order logic to talk about strings and their meanings. Each nonterminal symbol becomes a one-place predicate that is true of strings that are phrases of that category. For example, *Noun*("*stench*") is a true logical sentence, whereas *Noun*("*the*") is false. It is easy to write BNF rules as logical implication sentences in first-order logic:

BNF	**First-Order Logic**
$S \rightarrow NP\ VP$	$NP(s_1) \land VP(s_2) \;\Rightarrow\; S(Append(s_1, s_2))$
$Noun \rightarrow$ ***stench*** $\mid\ \ldots$	$(s = \text{"}stench\text{"} \lor \ldots) \;\Rightarrow\; Noun(s)$

The first of these rules says that if there is a string s_1 that is a noun phrase and a string s_2 that is a verb phrase, then the string formed by appending them together is a sentence. The second rule says that if s is the string "stench" (or one of the other words not shown), then the string s is a noun. A grammar written with logical sentences is called a **logic grammar**. Since unrestricted logical inference is computationally expensive, most logic grammars insist on a restricted format. The most common is **definite clause grammar** or **DCG**, in which every sentence must be a definite clause.[5]

LOGIC GRAMMAR

DEFINITE CLAUSE GRAMMAR

DCG

The DCG *formalism* is attractive because it allows us to describe grammars in terms of something we understand well: first-order logic. Unfortunately, it has the disadvantage of being more verbose than BNF notation. We can have the best of both by defining a special DCG *notation* that is an extension of BNF, but retains the well-founded DCG semantics. From now on, when we say "definite clause grammar," we mean a grammar written in this special notation, which is defined as follows:

- The notation $X \rightarrow Y\ Z\ \ldots$ translates as $Y(s_1) \land Z(s_2) \land \ldots \;\Rightarrow\; X(Append(s_1, s_2, \ldots)$.
- The notation $X \rightarrow$ ***word*** translates as $X([\text{"}word\text{"}])$.
- The notation $X \rightarrow Y \mid Z \mid \ldots$ translates as $Y'(s) \lor Z'(s) \lor \ldots \;\Rightarrow\; X(s)$, where Y' is the translation into logic of the DCG expression Y.

These three rules allow us to translate BNF into definite clauses. We now see how to extend the notation to incorporate grammars that can not be expressed in BNF.

- Nonterminal symbols can be **augmented** with extra arguments. In simple BNF notation, a nonterminal such as *NP* gets translated as a one-place predicate where the single argument represents a string: $NP(s)$. In the augmented DCG notation, we can write $NP(sem)$, for example, to express "an *NP* with semantics *sem*." This gets translated into logic as the two-place predicate $NP(sem, s)$.

[5] A definite clause is a type of Horn clause that, when written as an implication, has exactly one atom in its consequent, and a conjunction of zero or more atoms in its antecedent, for example, $A_1 \land A_2 \land \ldots \;\Rightarrow\; C_1$.