**EXPERIMENT - 8**

**Aim**: Sentence generation using context-free grammar

**Theory**:

**Context Free Grammar**

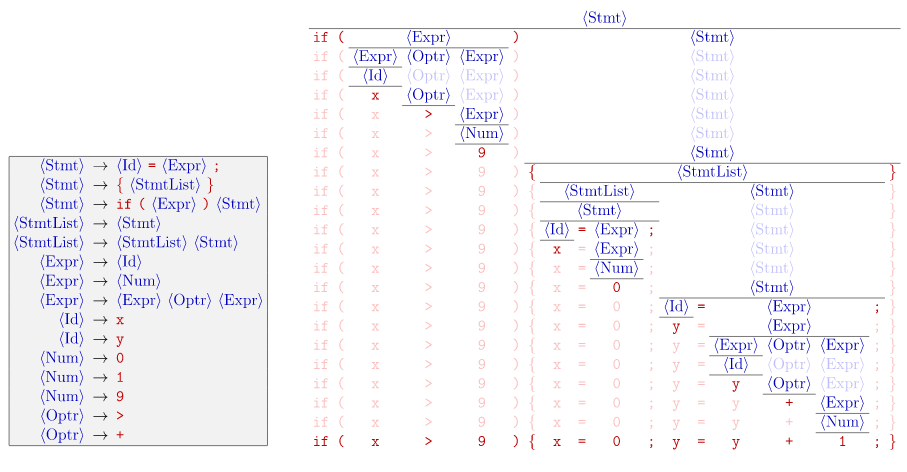
Context-free grammars (CFGs) are used to describe context-free languages. A context-free grammar is a set of recursive rules used to generate patterns of strings. A context-free grammar can describe all regular languages and more, but they cannot describe all possible languages.

Context-free grammar (CFG) is a formal grammar whose production rules can be applied to a nonterminal symbol regardless of its context. In a context-free grammar, each production rule is of the form.

with A a single nonterminal symbol, and α a string of terminals and/or nonterminal (α can be empty). A formal grammar is essentially a set of production rules that describe all possible strings in a given formal language. Production rules are simple replacements. For example, the first rule in the picture,

⟨Stmt⟩ → ⟨Id⟩ = ⟨Expr⟩

replaces ⟨Stmt⟩ with ⟨Id⟩ = ⟨Expr⟩ There can be multiple replacement rules for a given nonterminal symbol. The language generated by a grammar is the set of all strings of terminal symbols that can be derived, by repeated rule applications, from some particular nonterminal symbol ("start symbol"). Nonterminal symbols are used during the derivation process, but do not appear in its result string.



**Code**:

| from collections import defaultdict  import random  class CFG(object):  def \_\_init\_\_(self):  self.rules = defaultdict(list)  def add\_production\_rule(self, lhs, rhs):  productions = rhs.split('|')  for rules in productions:  self.rules[lhs].append(tuple(rules.split()))  def generate\_random(self, symbol):  sentence = ''  random\_production = random.choice(self.rules[symbol])  for sym in random\_production:  if sym in self.rules:  sentence += self.generate\_random(sym)  else:  sentence += sym + ' '  return sentence  cfg = CFG()  cfg.add\_production\_rule('S', 'NP VP')  cfg.add\_production\_rule('NP', 'Det N')  cfg.add\_production\_rule('PP', 'P NP')  cfg.add\_production\_rule('VP', 'V NP | V PP | VP')  cfg.add\_production\_rule('Det', 'the | a')  cfg.add\_production\_rule('N', 'man | elephant | dog | cat')  cfg.add\_production\_rule('P', 'in | with | on')  cfg.add\_production\_rule('V', 'slept | saw | walked | kicked | followed | shot')  print(dict(cfg.rules))  random\_sentences = []  for i in range(10):  sentence = cfg.generate\_random('S')  if sentence not in random\_sentences:  random\_sentences.append(sentence)  random\_sentences |
| --- |

**Output**:

| {'S': [('NP', 'VP')], 'NP': [('Det', 'N')], 'PP': [('P', 'NP')], 'VP': [('V', 'NP'), ('V', 'PP'), ('VP',)], 'Det': [('the',), ('a',)], 'N': [('man',), ('elephant',), ('dog',), ('cat',)], 'P': [('in',), ('with',), ('on',)], 'V': [('slept',), ('saw',), ('walked',), ('kicked',), ('followed',), ('shot',)]}  ['the elephant walked in a cat ',  'the man walked with a cat ',  'a man kicked a man ',  'a dog kicked with a man ',  'a elephant saw in a dog ',  'a dog saw with a cat ',  'a man walked with a elephant ',  'the elephant kicked in the elephant ',  'the elephant saw in the dog ',  'the elephant shot in a dog '] |
| --- |

**Conclusion**:

Context-free grammar provides a set of rules that define the generation of a sentence structure in a language. These sets of rules generate patterns of strings that satisfy the language rules. The rules are defined in terms of set of variables that are recursively defined in terms of another, the final rules thus do not require knowledge of context or symbols that may or not be present. Thus, these symbols can be substituted in terms of another leading to the CFG production rules.