Page 1: What is CFG?

CFG (Context-Free Grammar) is a way to describe the structure of a language using rules.

A CFG is defined as:

G = (V, Sigma, R, S)

- V -> Variables (Non-terminals)
- Sigma -> Alphabet (Terminals)
- R -> Rules or productions
- S -> Start symbol

Page 2: Example CFG

Lets define a simple CFG to generate balanced parentheses:

G = (V, Sigma, R, S)

- V = {S}
- Sigma = {(,)}
- R:
  - S -> (S)
  - S -> SS
  - S -> epsilon (empty string)

This grammar generates:

- ""
- ()
- ()()

- (())

- (()())

Page 3: Python Program to Simulate CFG

Well use the `lark` library to simulate CFG parsing.

Step 1: Install the library

pip install lark

Step 2: Python Code:

```
from lark import Lark

# Define a CFG for balanced parentheses
cfg_grammar = """
    start: expr

    expr: "(" expr ")"        -> parens
        | expr expr           -> concat
        |                     -> empty
"""

# Create a parser
parser = Lark(cfg_grammar, start='start')

# Test input strings
test_strings = ["", "()", "(())", "()()", "(()())", "((()", "())("]
```

```
for s in test_strings:
    try:
        parser.parse(s)
        print(f" '{s}' is VALID")
    except:
        print(f" '{s}' is INVALID")
```

Page 4: Output of the Program

```
 '' is VALID
 '()' is VALID
 '(())' is VALID
 '()()' is VALID
 '(()())' is VALID
 '(()' is INVALID
 '())(' is INVALID
```