SCS 4209 / IS 4108 / CS 4113 - Natural Language Processing
Practical 06

1. Set up the environment.

a. Install NLTK if not already installed using pip.

pip install nltk

b. Import the necessary NLTK modules such as CFG and ChartParser or RecursiveDescentParser.

import nltk: Imports the main NLTK module.

from nltk import CFG: Imports the CFG module for defining context-free grammars.

lets you define your own grammar rules for sentences

from nltk.parse import ChartParser, RecursiveDescentParser: Imports the parsers you want to use.

2. Define the Context Free Grammar (CFG)

a. Define your context-free grammar using the CFG.fromstring() function. [Ensure the CFG includes rules for sentences (S), noun phrases (NP), verb phrases (VP), prepositional phrases (PP), verbs (V), nouns (N), and determiners (Det)]

```
# Define the context-free grammar
cfg = CFG.fromstring("""
    S -> NP VP
    VP -> V NP | VP PP
    PP -> P NP
    NP -> 'I' | Det N | Det N PP
    V -> 'saw' | 'ate' | 'walked'
    N -> 'man' | 'hill' | 'telescope' | 'dog' | 'cat' | 'park'
    Det -> 'a' | 'A' | 'the' | 'my'
    P -> 'on' | 'with' | 'by' | 'in'
""")
```

The parser only recognizes these specific words as noun, verbs, determinants or prepositions because those are the terminals you declared in your grammar.

If you try to parse a sentence with a word not in your grammar, for example:

"the bird chased the dog"
The word "bird" is not defined in N -> 'cat' | 'dog', so the parser will fail to parse it.

Parsing a sentence means analyzing its structure according to grammar rules to understand how the parts of the sentence relate to each other.

So, when you "parse a sentence" with a tool like NLTK, you feed it the sentence and your grammar, and it outputs this structured tree showing how the sentence parts fit together.

b. Create a chart parser with the defined grammar.

```
parser = ChartParser(cfg)
```

It can return multiple parse trees if the sentence can be interpreted in more than one way.

but, all the parse trees that the parser builds come strictly from the grammar rules you defined.

3. Parse a sentence

a. Create a sentence to be parsed
"I saw the man with the telescope"

b. Split the sentence into a list of words
We can split the sentence into a list of words using Python's split() method.

c. Parse the sentence using your CFG and the ChartParser
We'll use the ChartParser to parse the sentence based on the previously defined CFG.

d. Print out the parse tree(s)

```
"I saw the man with the telescope"

words = sentence.split()

print(f"Parsing sentence: {sentence}")
for tree in parser.parse(words):
        print(tree)
        tree.pretty_print()
```

4. Modify the CFG

To reduce ambiguity in the CFG, we can refine the grammar rules to be more specific. Ambiguity often arises from rules that are too general and can apply to multiple structures. In our original CFG, the ambiguity might come from the fact that prepositional phrases (PP) can attach to different parts of the sentence (e.g., to the verb phrase or to the noun phrase).

Modify your CFG to reduce ambiguity

```
import nltk
from nltk import CFG
from nltk.parse import ChartParser

# Define the modified context-free grammar to reduce ambiguity
cfg = CFG.fromstring("""
    S -> NP VP
    VP -> V NP | V NP PP
    PP -> P NP
    NP -> 'I' | Det N | Det N PP
    V -> 'saw' | 'ate' | 'walked'
    N -> 'man' | 'hill' | 'telescope' | 'dog' | 'cat' | 'park'
    Det -> 'a' | 'A' | 'the' | 'my'
    P -> 'on' | 'with' | 'by' | 'in'
""")
```

```
# Define the context-free grammar
cfg = CFG.fromstring("""
S-> NPVP
VP-> V NP|VPPP
PP-> P NP
NP-> 'I' | Det N | Det N PP
V-> 'saw' | 'ate' | 'walked'
N-> 'man' | 'hill' | 'telescope' | 'dog' | 'cat' | 'park'
Det-> 'a' | 'A' | 'the' | 'my'
P-> 'on' | 'with' | 'by' | 'in'
""")
```

```
# Create a chart parser with the defined grammar
parser = ChartParser(cfg)

# a. Create a sentence to be parsed
sentence = "I saw the man with the telescope"

# b. Split the sentence into a list of words
words = sentence.split()

# c. Parse the sentence using your CFG and the ChartParser
print(f"Parsing sentence: {sentence}")
for tree in parser.parse(words):
    # d. Print out the parse tree(s)
    print(tree)
    tree.pretty_print()
```

5. Define the Context Free Grammar (CFG) in Sinhala

```
import nltk
from nltk import CFG
from nltk.parse import ChartParser

# Define the context-free grammar in Sinhala
cfg = CFG.fromstring("""
    S -> NP VP
    VP -> V NP | V NP PP
    PP -> P NP
    NP -> 'මම' | Det N | Det N PP
    V -> 'යනවා' | 'කනවා' | 'බලනවා' | 'ඉගනගන්නවා'
    N -> 'පාසල' | 'ගුරුවරයා' | 'පොත' | 'දුරදක්න' | 'බල්ල' | 'පූසා' | 'උද්යානය'
    Det -> 'එක' | 'මමේ' | 'ඒ' | 'මගේ'
    P -> 'ට' | 'සමග' | 'අසල' | 'දී'
""")

# Create a chart parser with the defined grammar
parser = ChartParser(cfg)

# Create a sentence to be parsed
sentence = "මම පාසලට යනවා"

# Split the sentence into a list of words
words = sentence.split()

# Parse the sentence using your CFG and the ChartParser
print(f"Parsing sentence: {sentence}")
for tree in parser.parse(words):
    # Print out the parse tree(s)
    print(tree)
    tree.pretty_print()
```

Exercise

Example Code:

```
import nltk
from nltk import CFG
from nltk.parse import ChartParser

# Define the context-free grammar in English
cfg = CFG.fromstring("""
    S -> NP VP
    VP -> V NP | V NP PP
    PP -> P NP
    NP -> 'She' | Det N | Det N PP
    V -> 'found' | 'saw' | 'walked'
    N -> 'dog' | 'cat' | 'park' | 'book' | 'telescope'
    Det -> 'a' | 'the' | 'her'
    P -> 'in' | 'with' | 'at'
""")

# Create a chart parser with the defined grammar
parser = ChartParser(cfg)

# Create a sentence to be parsed
sentence = "She found the book in the park"

# Split the sentence into a list of words
words = sentence.split()

# Parse the sentence using the CFG and the ChartParser
print(f"Parsing sentence: {sentence}")
for tree in parser.parse(words):
    # Print out the parse tree(s)
    print(tree)
    tree.pretty_print()
```

Questions

1. Modify the CFG to handle additional sentences such as "She saw a dog with a telescope" and "Her cat walked in the park". Define the necessary rules and parse these sentences.
2. Analyze if there are any ambiguities in the parse trees generated by the CFG. Explain how you can modify the CFG to reduce ambiguity.
3. Create your own sentences and see if they can be parsed using the defined CFG. Adjust the CFG rules if necessary to accommodate new sentences.