

Shift Reducer Parser

```
import nltk
from nltk.grammar import CFG
from nltk.parse.shiftreduce import ShiftReduceParser

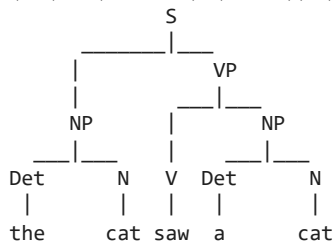
# Define a simple context-free grammar
grammar = CFG.fromstring("""
S -> NP VP
NP -> Det N
VP -> V NP
Det -> 'the' | 'a'
N -> 'cat' | 'dog'
V -> 'chased' | 'saw'
""")

# Create the Shift-Reduce Parser
parser = ShiftReduceParser(grammar)

# Example sentence (tokenized)
sentence = ['the', 'cat', 'saw', 'a', 'cat']

# Parsing the sentence
for tree in parser.parse(sentence):
    print(tree)
    tree.pretty_print()
```

(S (NP (Det the) (N cat)) (VP (V saw) (NP (Det a) (N cat)))))



Recursive Parser

```
tokens = ['a', 'dog', 'see', 'the', 'cat']
pos = 0

def match(expected):
    global pos
    if pos < len(tokens) and tokens[pos] == expected:
        pos += 1
        return True
    return False

def parseS():
    return parseNP() and parseVP()

def parseNP():
    return parseDet() and parseN()

def parseVP():
    return parseV() and parseNP()

def parseDet():
    return match('the') or match('a')

def parseN():
    return match('dog') or match('cat')

def parseV():
```

```
        return match('chased') or match('saw')

if parseS() and pos == len(tokens):
    print('Sentence is valid according to grammar.')

import nltk

grammar = nltk.CFG.fromstring("""
    S -> NP VP
    NP -> Det N
    VP -> V NP
    Det -> 'a' | 'the'
    N -> 'dog' | 'cat'
    V -> 'chased' | 'saw'
""")

parser = nltk.ChartParser(grammar)
trees = list(parser.parse(tokens))

if trees:
    for tree in trees:
        print(tree)
        trees[0].pretty_print()
    else:
        print("No parse trees found for the given sentence with the current grammar.")
else:
    print('Sentence is invalid.')
```

Sentence is invalid.