In [11]:

```
import nltk
from nltk import CFG
from nltk.parse import EarleyChartParser

pcfg_grammar = CFG.fromstring("""
    S -> NP VP
    NP -> Det N  | NP PP  | 'John'
    Det -> 'the'  | 'a'
    N -> 'man'  | 'dog'  | 'cat'
    VP -> V NP | VP PP
    V -> 'chased'  | 'saw'
    PP -> P NP
    P -> 'with'  | 'in'
""")

def probabilistic_parsing(sentence):
    tokens = nltk.word_tokenize(sentence)
    parser = EarleyChartParser(pcfg_grammar)
    for tree in parser.parse(tokens):
        print("Parse Tree with Probability:", tree)
        break
example_sentence = "the man saw a cat with a dog"
probabilistic_parsing(example_sentence)
```

```
Parse Tree with Probability: (S
  (NP (Det the) (N man))
  (VP
    (VP (V saw) (NP (Det a) (N cat)))
    (PP (P with) (NP (Det a) (N dog)))))
```

In [12]:

```
import nltk
from nltk import PCFG
from nltk.parse import EarleyChartParser

pcfg_grammar = PCFG.fromstring("""
    S -> NP VP [1.0]
    NP -> Det N [0.5] | NP PP [0.4] | 'John' [0.1]
    Det -> 'the' [0.6] | 'a' [0.4]
    N -> 'man' [0.5] | 'dog' [0.3] | 'cat' [0.2]
    VP -> V NP [0.7] | VP PP [0.3]
    V -> 'chased' [0.4] | 'saw' [0.6]
    PP -> P NP [1.0]
    P -> 'with' [0.7] | 'in' [0.3]
""")

def probabilistic_parsing(sentence):
    tokens = nltk.word_tokenize(sentence)
    parser = EarleyChartParser(pcfg_grammar)
    for tree in parser.parse(tokens):
        print("Parse Tree with Probability:", tree)
        break  # Exit the loop after printing the first parse tree

# Example sentence
example_sentence = "the man saw a cat with a dog"

# Perform probabilistic parsing
probabilistic_parsing(example_sentence)
```

```
Parse Tree with Probability: (S
  (NP (Det the) (N man))
  (VP
    (VP (V saw) (NP (Det a) (N cat)))
    (PP (P with) (NP (Det a) (N dog)))))
```

In [13]:

```
import nltk
from nltk.corpus import treebank
from nltk.tag import DefaultTagger, UnigramTagger, BigramTagger, TrigramTagger
from nltk.chunk import RegexpParser

nltk.download('treebank')

def train_pos_tagger():
    tagged_sentences = treebank.tagged_sents()
    train_data = tagged_sentences[:3000]
    default_tagger = DefaultTagger('NN')
    unigram_tagger = UnigramTagger(train_data, backoff=default_tagger)
    bigram_tagger = BigramTagger(train_data, backoff=unigram_tagger)
    trigram_tagger = TrigramTagger(train_data, backoff=bigram_tagger)
    accuracy = trigram_tagger.evaluate(tagged_sentences[3000:])
    print("Trigram Tagger Accuracy:", accuracy)
    return trigram_tagger

def parse_sentence(tagged_sentence):
    chunking_grammar = r"""
        NP: {<DT>?<JJ>*<NN>}
        VP: {<VB.*><NP|PP|CLAUSE>+$}
        CLAUSE: {<NP><VP>}
    """
    chunk_parser = RegexpParser(chunking_grammar)
    tree = chunk_parser.parse(tagged_sentence)
    print("\nParse Tree:")
    tree.pretty_print()

pos_tagger = train_pos_tagger()

example_sentence = "The quick brown fox jumps over the lazy dog."

tokenized_sentence = nltk.word_tokenize(example_sentence)
tagged_sentence = pos_tagger.tag(tokenized_sentence)

print("\nTagged Sentence:", tagged_sentence)

parse_sentence(tagged_sentence)
```

```
[nltk_data] Downloading package treebank to /home/cmr/nltk_data...
[nltk_data]   Package treebank is already up-to-date!
/tmp/ipykernel_4177/4065806774.py:15: DeprecationWarning:
  Function evaluate() has been deprecated.  Use accuracy(gold)
  instead.
  accuracy = trigram_tagger.evaluate(tagged_sentences[3000:])

Trigram Tagger Accuracy: 0.8806388948845241

Tagged Sentence: [('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('f
ox', 'NN'), ('jumps', 'NN'), ('over', 'IN'), ('the', 'DT'), ('lazy',
'NN'), ('dog', 'NN'), ('.', '.')]

Parse Tree:
                              S
    ___________________________|_____________________________________
____
   |     |            NP               NP      NP            NP
```

```
NP
   |     |     _______|________         |       |        _____|_____
|
over/IN ./. The/DT quick/JJ brown/NN fox/NN jumps/NN the/DT     lazy/N
N dog/NN
```

In [ ]: