```
cfg_rules = """
S -> NP VP
NP -> Det N | PropN
Det -> PosPro | Art
VP -> Vt NP

Art -> 'the' | 'a'
PropN -> 'Alice'
N -> 'duck' | 'telescope' | 'park'
Vt -> 'saw'
PosPro -> 'my' | 'her'
"""
```

```
import nltk
```

```
cfg = nltk.CFG.fromstring(cfg_rules)
```

```
cfg.is_flexible_chomsky_normal_form()
```

```
True
```

```
cfg.start()
```

```
S
```

```
cfg.productions()
```

```
[S -> NP VP,
 NP -> Det N,
 NP -> PropN,
 Det -> PosPro,
 Det -> Art,
 VP -> Vt NP,
 Art -> 'the',
 Art -> 'a',
 PropN -> 'Alice',
 N -> 'duck',
 N -> 'telescope',
 N -> 'park',
 Vt -> 'saw',
 PosPro -> 'my',
 PosPro -> 'her']
```

```
# is
cfg.chomsky_normal_form()
```

```
<Grammar with 17 productions>
```

```
# sentences = [
#     "the purchase price includes two ancillary companies .".split(),
#     "the guild began a strike against the TV and movie industry in March 1988 .'
# ]
# for s in sentences:
#     grammar.check_coverage(s)

print('''
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-49a3a8a0a0b8> in <module>()
      4 ]
      5 for s in sentences:
----> 6     grammar.check_coverage(s)

/usr/local/lib/python3.7/dist-packages/nltk/grammar.py in check_coverage(self, tok
    664             missing = ", ".join(f"{w!r}" for w in missing)
    665             raise ValueError(
--> 666                 "Grammar does not cover some of the " "input words: %r." %
    667             )
    668 

ValueError: Grammar does not cover some of the input words: "'the', 'purchase', 'p
''')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-49a3a8a0a0b8> in <module>()
      4 ]
      5 for s in sentences:
----> 6     grammar.check_coverage(s)

/usr/local/lib/python3.7/dist-packages/nltk/grammar.py in check_coverage(self, tokens)
    664             missing = ", ".join(f"{w!r}" for w in missing)
    665             raise ValueError(
--> 666                 "Grammar does not cover some of the " "input words: %r." % missi
    667             )
    668 

ValueError: Grammar does not cover some of the input words: "'the', 'purchase', 'price',
############################################################################
```

```
from nltk import CFG


grammar = CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | NP PP
VP -> V NP | VP PP
Det -> 'a' | 'the'
N -> 'dog' | 'cat'
V -> 'chased' | 'sat'
P -> 'on' | 'in'
""")
```

```
grammar
```

```
    <Grammar with 14 productions>
```

```
grammar.start()
```

```
    S
```

```
grammar.productions()
```

```
    [S -> NP VP,
     PP -> P NP,
     NP -> Det N,
     NP -> NP PP,
     VP -> V NP,
     VP -> VP PP,
     Det -> 'a',
     Det -> 'the',
     N -> 'dog',
     N -> 'cat',
     V -> 'chased',
     V -> 'sat',
     P -> 'on',
     P -> 'in']
```

```
from nltk import PCFG
toy_pcfg1 = PCFG.fromstring("""
S -> NP VP [1.0]
NP -> Det N [0.5] | NP PP [0.25] | 'John' [0.1] | 'I' [0.15]
Det -> 'the' [0.8] | 'my' [0.2]
N -> 'man' [0.5] | 'telescope' [0.5]
VP -> VP PP [0.1] | V NP [0.7] | V [0.2]
V -> 'ate' [0.35] | 'saw' [0.65]
PP -> P NP [1.0]
P -> 'with' [0.61] | 'under' [0.39]
""")
```

```
g = CFG.fromstring("VP^<TOP> -> VBP NP^<VP-TOP>")
g.productions()[0].lhs()
```

```
VP^<TOP>
```

```
from nltk.grammar import CFG
from nltk.parse.generate import generate
grammar = CFG.fromstring("""
S -> A B
A -> 'a'
# An empty string:
B -> 'b' | ''
""")
list(generate(grammar))
```

```
[['a', 'b'], ['a', '']]
```

```
grammar = CFG.fromstring("""
S -> A B
A -> 'a'
# An empty production:
B -> 'b' |
""")
list(generate(grammar))
```

```
[['a', 'b'], ['a']]
```

```
str = ["a".split()]
for s in str:
  a = grammar.check_coverage(s)
  print(a)
```

```
None
```

```
grammar.productions()[0].lhs()
```

```
S
```

```
cnf_grammar = cfg.chomsky_normal_form()
```

```
##########################################################################
#########
```

```
grammar1 = nltk.CFG.fromstring("""
S -> NP VP
VP -> V NP | V NP PP
PP -> P NP
V -> "saw" | "ate" | "walked"
NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
N -> "man" | "dog" | "cat" | "telescope" | "park"
P -> "in" | "on" | "by" | "with"
""")
```

```
sent = "John ate my telescope".split()
rd_parser = nltk.RecursiveDescentParser(grammar1)
for tree in rd_parser.parse(sent):
     print(tree)
```

```
(S (NP John) (VP (V ate) (NP (Det my) (N telescope))))
```

```
cnf_grammar = grammar1.chomsky_normal_form()
```

```
from nltk.parse.chart import BottomUpChartParser
parser = BottomUpChartParser(cnf_grammar)
parses = list(parser.parse(sent))
```

```
parses
```

```
[Tree('S', [Tree('NP', ['John']), Tree('VP', [Tree('V', ['ate']), Tree('NP',
[Tree('Det', ['my']), Tree('N', ['telescope'])])])])]
```

```
from nltk import CFG
grammar = CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | NP PP
VP -> V NP | VP PP
Det -> 'a' | 'the'
N -> 'dog' | 'cat'
V -> 'chased' | 'sat'
P -> 'on' | 'in'
""")
```

```
sent0 = "dog chased a dog".split()
```

```
cnf_grammar0 = grammar.chomsky_normal_form()
```

```
from nltk.parse.chart import BottomUpChartParser
parser = BottomUpChartParser(cnf_grammar0)
parses = list(parser.parse(sent0))
```

```
parses
```

```
    []
```