

```
In [16]: from grammar import *
         from parser import *
         from util2 import *
```

## Our Grammar

The current grammar:

```
In [17]: print str(timeFliesPCFG2)

Noun => flies      | 0.5
Noun => arrow      | 0.4
Noun => time       | 0.1
TOP  => S          | 1.0
Det  => an         | 1.0
VP   => Verb NP    | 0.1
VP   => Verb PP    | 0.1
VP   => Verb       | 0.1
VP   => Verb NP_PP | 0.7
S    => VP | 0.7
S    => NP VP      | 0.1
S    => VP PP      | 0.1
S    => NP VP_PP   | 0.1
VP_PP => VP PP    | 1.0
NP_PP => NP PP    | 1.0
Prep => like      | 1.0
PP   => Prep NP   | 1.0
Verb => flies     | 0.4
Verb => like      | 0.2
Verb => time      | 0.4
NP   => Det Noun  | 0.7
NP   => Noun      | 0.3
```

Just the unary rules:

```
In [18]: for lhs in timeFliesPCFG2.pcfgC.iterkeys():
         for rhs,count in timeFliesPCFG2.pcfgC[lhs].iteritems():
             rule = Rule(lhs,rhs)
             if rule.isUnary:
                 print rule

Noun => flies
Noun => arrow
Noun => time
TOP  => S
Det  => an
VP   => Verb
S    => VP
Prep => like
Verb => flies
Verb => like
Verb => time
NP   => Noun
```

# A Simple Example

Chart on a simple example.

```
In [19]: simpleSent = ['time', 'flies']
chart = cky(timeFliesPCFG2, simpleSent, pruningPercent=None)
printChart(chart, simpleSent, widths=(5,11,7,3,2), printBackPointers=True, printProbs=True)
```

row0				
1				
Noun	time[0:1]	None	-2.30	
TOP	S[0:1]	None	-3.58	
VP	Verb[0:1]	None	-3.22	
S	VP[0:1]	None	-3.58	
Verb	time[0:1]	None	-0.92	
time	None	None	0.00	
NP	Noun[0:1]	None	-3.51	
2				
	VP	Verb[0:1]	NP[1:2]	-5.12
	S	VP[0:2]	None	-5.47
	TOP	S[0:2]	None	-9.03
row1				
1				
2				
	flies	None	None	0.00
	Noun	flies[1:2]	None	-0.69
	TOP	S[1:2]	None	-3.58
	VP	Verb[1:2]	None	-3.22
	S	VP[1:2]	None	-3.58
	Verb	flies[1:2]	None	-0.92
	NP	Noun[1:2]	None	-1.90

The parse:

```
In [20]: print parse(timeFliesPCFG2, simpleSent)

(TOP: (S: (VP: (Verb: 'time') (NP: (Noun: 'flies')))))
```

FYI, 'TOP' is pointing to 'S'

```
In [21]: N = len(simpleSent)
top = chart.best_in_cell(0,N,'TOP')
printItem(top)
s = top.backPtrLeft
printItem(s)
```

		TOP[0:2]	
Left:	S[0:2]		
Right:	None		
Prob:	-9.02801881518		
Tree:	(TOP: (S: (VP: (Verb: 'time') (NP: (Noun: 'flies')))))		
		S[0:2]	
Left:	VP[0:2]		
Right:	None		
Prob:	-5.47267075369		
Tree:	(S: (VP: (Verb: 'time') (NP: (Noun: 'flies'))))		

# On the Full Sentence

Desired final output:

```
In [22]: print desiredTimeFliesParse
(TOP:
  (S:
    (VP:
      (Verb: 'time')
      (NP: (Noun: 'flies'))
      (PP: (Prep: 'like') (NP: (Det: 'an') (Noun: 'arrow'))))))
```

Actual output:

```
In [23]: myTree = parse(timeFliesPCFG2, timeFliesSent)
print myTree
(TOP:
  (S:
    (VP:
      (Verb: 'time')
      (NP_PP:
        (NP: (Noun: 'flies'))
        (PP: (Prep: 'like') (NP: (Det: 'an') (Noun: 'arrow'))))))))
```

Successful evaluation:

```
In [24]: print evaluate(desiredTimeFliesParse, myTree)
1.0
```

The chart:

```
In [25]: chart = cky(timeFliesPCFG2, timeFliesSent, pruningPercent=None)
printChart(chart, timeFliesSent, widths=(5,11,7,1,2), printBackPointers=True, printProbs=True)
```

row0				
1				
Noun	time[0:1]	None	-2.30	
TOP	S[0:1]	None	-3.58	
VP	Verb[0:1]	None	-3.22	
S	VP[0:1]	None	-3.58	
Verb	time[0:1]	None	-0.92	
time	None	None	0.00	
NP	Noun[0:1]	None	-3.51	
2				
	VP	Verb[0:1]	NP[1:2]	-5.12
	S	VP[0:2]	None	-5.47
	TOP	S[0:2]	None	-9.03
3				
4				
5				
	VP	Verb[0:1]	NP_PP[1:5]	-4.44
	S	VP[0:5]	None	-4.80



