In [1]:
```
from grammar import *
from parser import *
from util2 import *
```

The current grammar:

In [2]:
```
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 [3]:
```
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
```

Chart on a simple example:

In [11]:
```
simpleSent = ['time','flies']
chart = cky(timeFliesPCFG2, simpleSent, pruningPercent=None)
printChart(chart, simpleSent, 20, False)
print parse(timeFliesPCFG2, simpleSent)
```

```
row0_____________________________________________________________
                    1
                      Noun
                       TOP
                        VP
                         S
                      Verb
                      time
                        NP
                                        2
                                            VP
                                             S
                                           TOP
row1_____________________________________________________________
                    1
                                        2
                                         flies
                                          Noun
                                           TOP
                                            VP
                                             S
                                          Verb
                                            NP
(TOP: (S: (NP: (Noun: 'time')) (VP: (Verb: 'flies'))))
```

Desired final output:

In [5]:
```
print desiredTimeFliesParse
```

```
(TOP:
  (S:
    (VP:
      (Verb: 'time')
      (NP: (Noun: 'flies'))
      (PP: (Prep: 'like') (NP: (Det: 'an') (Noun: 'arrow'))))))
```

Actual output:

In [6]:
```
myTree = parse(timeFliesPCFG2, timeFliesSent)
print myTree
print evaluate(desiredTimeFliesParse, myTree)
```

```
(TOP:
  (S:
    (VP: (Verb: 'time') (NP: (Noun: 'flies')))
    (PP: (Prep: 'like') (NP: (Det: 'an') (Noun: 'arrow')))))
0.9375
```

In [6]: