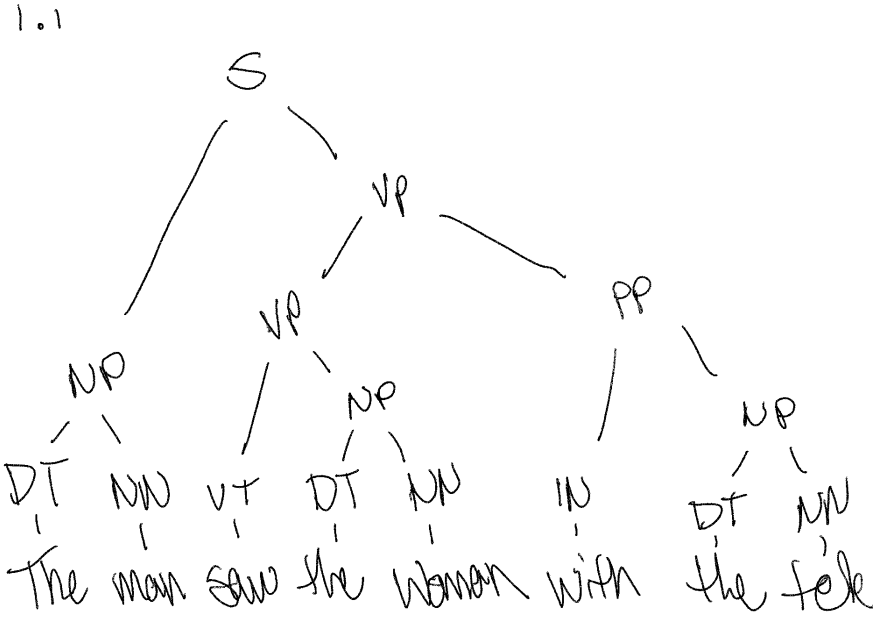
CS7347 HW3, Noah Gardner, 000843905

1 Assignment 3

1. Given the CFG grammar below. [**Points 30**]

S	→	NP VP	1.0
VP	→	Vi	0.4
VP	→	Vt NP	0.4
VP	→	VP PP	0.2
NP	→	DT NN	0.3
NP	→	NP PP	0.7
PP	→	IN NP	1.0
Vi	→	'sleeps'	1.0
Vt	→	'saw'	1.0
NN	→	'man'	0.7
NN	→	'woman'	0.2
NN	→	'telescope'	0.1
DT	→	'the'	1.0
IN	→	'with'	0.5
IN	→	'in'	0.5

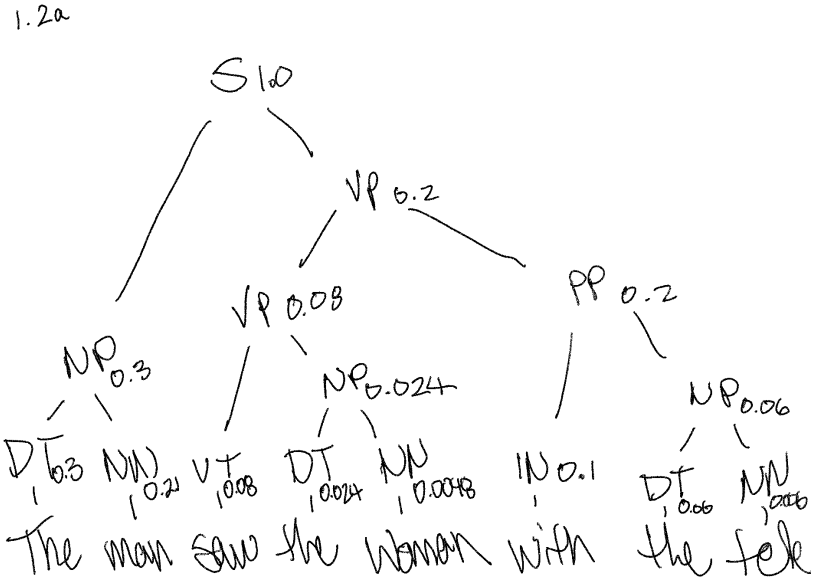
   (a) Parse the sentence using the above grammar and show parse tree. [**Points 10**]
   `The man saw the woman with the telescope.`
   Answer:



(b) Show probabilisitc parse tree with probability at each node in the tree for the following sentences. [**Points 20**]

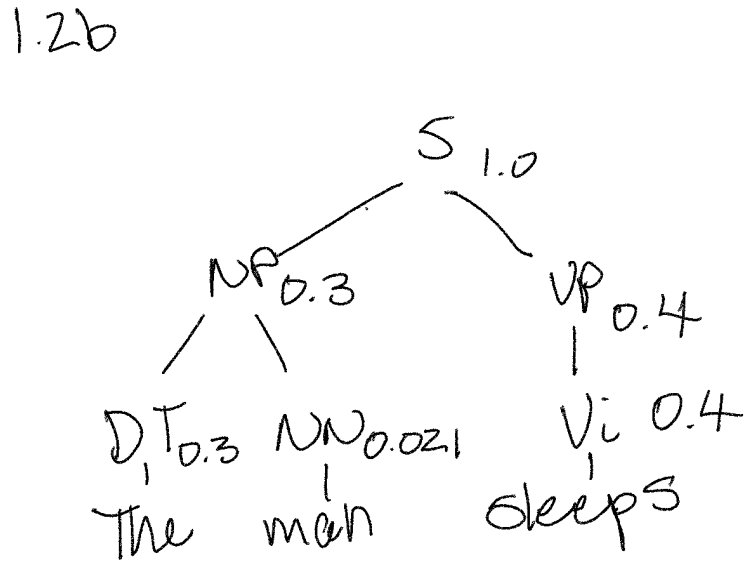
`The man saw the woman with the telescope.`

Answer:

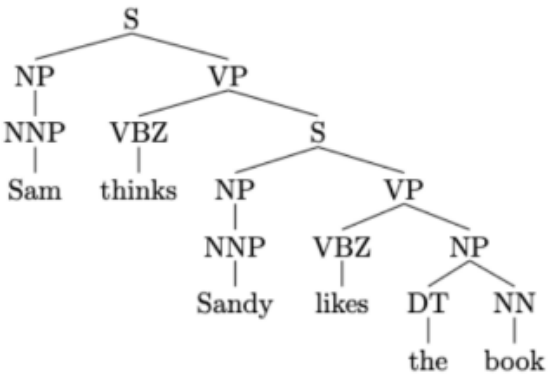


`The man sleeps.`

Answer:



2. Show the bracketed notation for the following tree. [**Points 10**]



Answer:

2

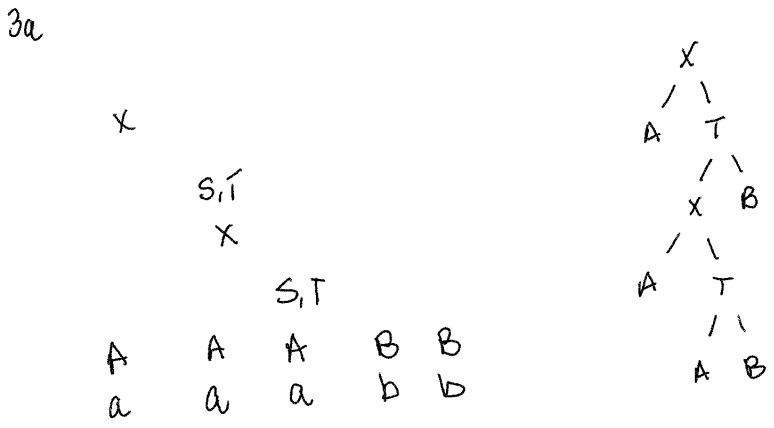
S(NP(NNP(Sam)))(VP(VBZ(thinks))(
S(NP(NNP(Sandy)))(VP(VBZ(likes))(
NP(DT(the))(NN(book)))))

3. Consider the grammar G given by:

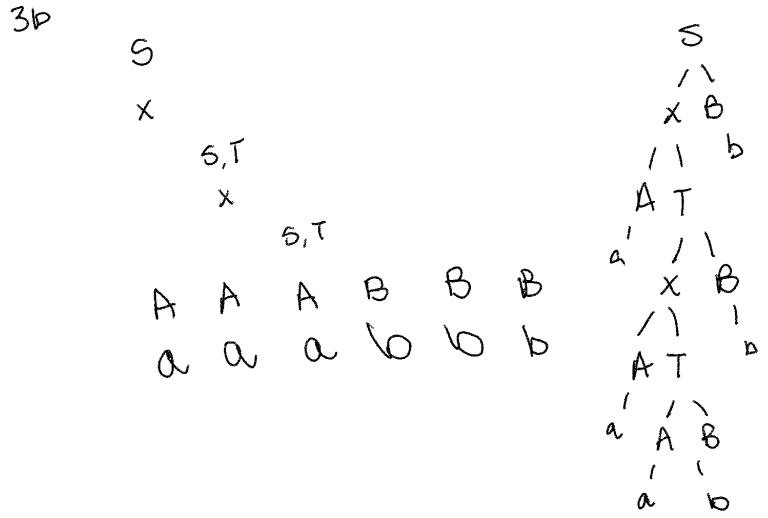
S	→	XB
T	→	AB — XB
X	→	AT
A	→	a
B	→	b

Validate your answer by using CKY algorithm. [**Points 20**]

(a) Is $w = aaabb$ in L(G)? [**Points 7.5**]
Answer:



(b) Is $w = aaabbb$ in L(G)? [**Points 7.5**]
Answer:



4. How are strings stored in spacy? Describe the document similarity checking mechanism in spacy? How would you compare similarity between two documents using spacy? Please write down an example code for document similarity checking. What are the spacy pipeline component initiated while using default nlp(). How would you add a custom component in the spacy nlp pipeline? Write a sample code which utilizes only two component tokenizer and add custom component in npl pipeline. **[Points 20]**

   Answer: Strings in spacy are stored as Documents. Document similarity uses a vector based model to compare two documents.

   Document similarity:

```
import spacy

nlp = spacy.load('en_core_web_md')
d1 = nlp('document 1')
d2 = nlp('document 2')
print(d1.similarity(d2))
```

   The default spacy pipeline components are tokenizer and parser. This is how to create a custom component. In this example, the custom component will delete extra tokens past 10.

```
import spacy
from spacy.language import Language

@Language.component('cutoff_component')
def cutoff_component(doc):
    if len(doc) > 10:
        return doc[:10]
    return doc

nlp = spacy.load('en_core_web_sm')
nlp.add_pipe(cutoff_component)
doc = nlp('This is a sample document')
```

5. Go through the sequence of transitions needed for parsing the sentence:
   `I parsed this sentence correctly`.
   The dependency tree is shown below. **[Points 25]**

(a) At each step, give the configuration of the stack and buffer, as well as what transition was applied this step and what new dependency was added (if any). **[Points 15]**

Answer: See table on next page.

(b) A sentence containing n words will be parsed in how many steps (in terms of n)? Briefly explain why. **[Points 10]**

Answer: The sentence will be parsed in $2n$ steps, because each word requires a step to push onto the stack, and a step to pop from the stack.

Step	Stack	Buffer	Transition	Dependency
0	[root]	[I, parsed, this, sentence, correctly]	shift(I)	
1	[root, I]	[parsed, this, sentence, correctly]	shift(parsed)	
2	[root, I, parsed]	[this, sentence, correctly]	leftarc	parsed $\rightarrow$ I
3	[root, parsed]	[this, sentence, correctly]	shift(this)	
4	[root, parsed, this]	[sentence, correctly]	shift(sentence)	
5	[root, parsed, this, sentence]	[correctly]	leftarc	this $\leftarrow$ sentence
6	[root, parsed, sentence]	[correctly]	rightarc	parsed $\rightarrow$ sentence
7	[root, parsed]	[]	shift(correctly)	
8	[root, parsed, correctly]	[]	rightarc	parsed $\rightarrow$ correctly
9	[root, parsed]	[]	rightarc	root $\rightarrow$ parsed