Name: _____

Section:_____

Linguistics 001
Spring 2009
Homework 8
Due: Wed., April 22nd for 15 points

PART I. CFG's and Chomsky Normal Form
The close relationship between modern linguistics and computer science is no accident. These grammars are instances of the literal, "input/output" conceptualization of production and comprehension that computer scientists and linguists share.

We've worked with two types of formal grammars so far this course. They involve surface forms and more abstract forms. You've learned to implement rules that "rewrite" abstract forms as surface forms.

Here's how rewrite rules work. Context-Free rules like (i) and (ii) below apply everywhere, rewriting the abstract symbol on the left-hand side as the abstract symbol or surface form on the right. Context-Sensitive rules like (iii) only apply in specific environments, which are either described or listed.

(i) N_{plural} ➙ N $affix_{plural}$
(ii) N ➙ {dæd , bɛt , ʃip , ... }
(iii) $affix_{plural}$ ➙ -z / [voiced] __
-s / [unvoiced] __
-∅ / { ʃip, dir, fiʃ }

1. **Translate the words below into their surface phonological forms (in IPA). Then, use the rules above to construct the morphological trees underlying the surface forms.**

/ / "dads"	/ / "sheep"	/ / "bets"

Below is a (simplified) categorization of Context-Free and Context-Sensitive Grammars.

	Input (left-hand side)	**Output (right-hand side)**
Context-Free Grammars (CFG)	Apply to all	zero or more nonterminal symbols and one or zero terminal symbols
Context-Sensitive Grammars (CSG)	Apply in specifically defined environment	zero or more nonterminal symbols and one or zero terminal symbols

2. **Identify each of the following as either a CFG or CSG. Give an example of a rule to support your answer.**

Syntax	**Phonology**
CFG or CSG	CFG or CSG
example rule:	example rule:

PART II. Normal forms and limitations

Below is a variant on the English "ABC song" that has traditionally been used to teach phonics. It is a call-and-response song wherein the teacher asks about a written letter and the children respond with the appropriate phonemes.

What does the A say?	/e/ /a/ /æ/
What does the B say?	/b/ /b/ /b/
What does the C say?	/s/ and /k/
What does the D say?	/d/ /d/ /d/
...	
What does the S say?	/s/ /s/ /s/
What does the T say?	/t/ /t/ /t/
...	
What does the X say?	/ks/ /ks/ /ks/
What does the Y say?	/jə/ /i/ /ai/

3. **In the space below, describe this excerpt of the song as a CFG which rewrites each letter as an appropriate sound. A few examples are included, to get you started. "A" is meant to represent the written form of a letter, and it's not important that it's capitalized. (ie, "A" = "a").**

S → {"A"or"B"or...or"Z"}	"B" →	"E" →	"Y" →
(ie, any single letter)	"C" →	"K" →	"Y" →
"A" →	"C" →	"S" →	"Y" →
"A" →	"D" →	"T" →	
"A" →	"E" →	"X" →	

4. **Which rule(s) above violate the output conditions of CFGs as listed in the above table?**

5. **If children use the "Phonics song" rewriting rules to map letters to sounds, how will they read the following words out loud? List all possibilities and put an asterisk (*) next to the incorrect forms.**

   "CATS"

   "CADS"

   "SAY"

PART III: Parsing and Dynamic Programming:

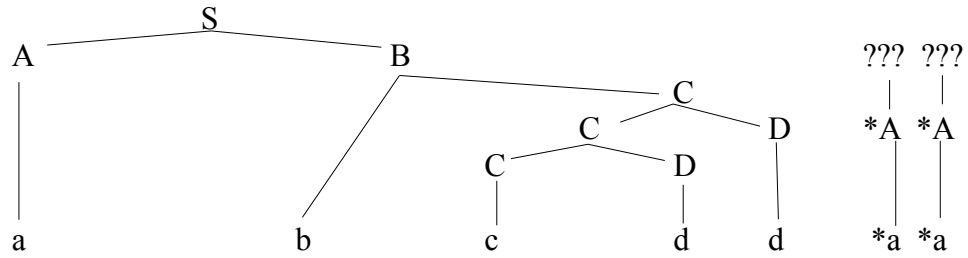
The Cocke Younger Kasami (CYK) algorithm analyzes strings and tells you whether they are grammatical (given a grammar that you yourself must design and supply). For instance, it would be able to tell you that the string in (i) could not be produced by the grammar in (ii):

(i) "abcddaa"

(ii)

S → AB	D → d
A → a	B → bC
C → c	C → CD

Now, if a person was asked to determine whether (i) could be produced by (ii), she would find it most convenient to do so by making a tree, like we do in syntax.

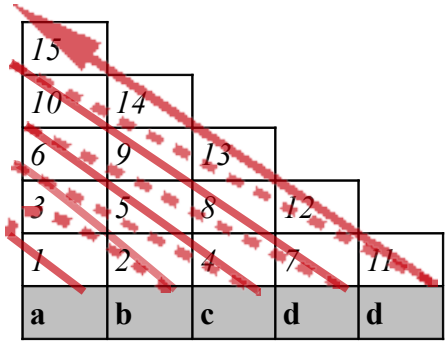


Clearly, (i) could not have been made using the grammar in (ii). No non-terminal will branch in such a way as to produce a structure with terminal *a*'s in those final positions.

Obviously, though, this is a huge pain. What if we wanted to do something more useful, like testing a grammar of English over a lot of random sentences? If we could get a computer to do it, we could look at lots of sentences and make sure that we're accurately describing English (even rare constructions).

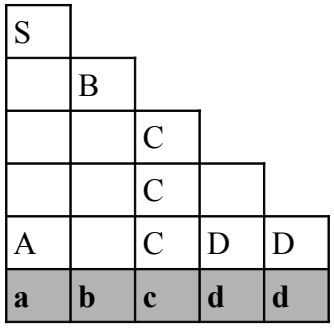
The problem is, computers don't understand trees like the one above. Hence, the development of algorithms which allow computers to decide whether or not a given string could have been produced by a given grammar. The CYK algorithm, demonstrated below, achieves this by doing a large number of tiny calculations, building tree structure from the ground up. Let's take an input that *IS* grammatical, according to (i):

Process:

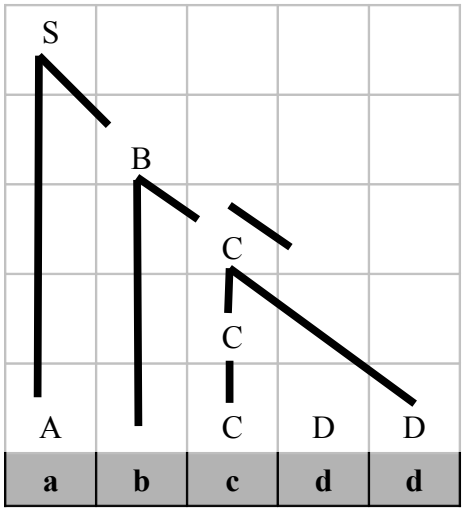


(the numbers and the arrows are both meant to show the same thing: the order in which the algorithm proceeds through the table)

Result:



Tree



This looks scary, but it isn't. Think of the process on the left as a (really boring) board game. Basically, what's happening is that, in each cell, starting in the lower left, you're asking, "How many of the things that are below and to the right of me can be captured using a single rule from the grammar?"

So starting in box 1...

1. Ask: "Reading from top to bottom, left to right, is there cell (or pair of cells)

   BELOW *(the box containing "a")*
   or
   BELOW AND TO THE RIGHT* *(the boxes containing "b","c","d", and "d")*

   of me that can be generated using a single rule?"

2. The first box below you is "a", and we CAN generate "a" with a single rule: A → a
3. Now, you can put the left hand side of that rule in the box. We have the first layer of our tree!
4. Repeat steps 1-3, adding more "left hand sides" to the same box if applicable.
5. When you can' t repeat steps 1-3 anymore, move on to the next box

...aaaaand, now we move on to the next box on the prim, rose path ;)
**Notice how step one does not consider boxes to the right of box 1 (eg. 4,7,11) or any boxes above it*

To extend this to a slightly more difficult case, Try to figure out how we generate the multiple C and D nodes (not a required homework question). Remember that you'll have to generate them in order!

Now, for the homework question.

6. **Refer the phrase structure rules that we used to do syntax earlier in the class. Forget about the "AND" rule for a second. It's just a hack anyway, and you might be able to see how it won't work with this algorithm. Prove that the string (a) is a grammatical sentence of English by filling out the table:**

"I will see the man with my new binoculars"

I	will	see	the	man	with	my	new	binoculars

S → NP TP	NP → N
TP → T VP	NP → det N
T → "will"	NP → NP PP
VP → V NP	N → A N
VP → VP PP	A → "*adjectives*"
V → "*verbs*"	PP → P NP
N → "*nouns*"	P → "*prepositions*"

For a gold star: how does structural ambiguity show up? If you're curious, see the Wikipedia article on the CYK Algorithm to find out how automatic (unsupervised) parsers can be trained to deal with it:

http://en.wikipedia.org/wiki/Stochastic_context-free_grammar

PART IV: Writing systems

As you heard Prof. Buckley tell you, almost all Chinese characters consist of two components: the semantic component or *radical*, and the *phonetic*, which is an approximation of the pronunciation. Those together are called one *character*. A famous linguist wrote that the Chinese writing system was *morphosyllabic*. That is, he observed that in Mandarin, most syllables are one morpheme and most morphemes are one syllable, and that most characters in Mandarin are also one syllable, and one morpheme.

1 character = 1 syllable = 1 morpheme

7. **Consider the following monomorphemic Mandarin words:**

zhāng láng
'cockroach'

pí pá
'flute'

Is the equation above violated? If so, how?

Many languages, including Mandarin, have a system of *diminuitivization*, in which attaching a morpheme to a word expresses smallness, comfort, and familiarity. In German, for instance, the word for “girl” is *Mäd-chen*, literally 'girl-little'. In Chinese, the morpheme *-r* 'boy' indicates this same meaning, and is used in words like “ball”, which counts as a single character and is pronounced as a single syllable.

< qiúr >
qiur
“ball"

8. What parts of this equation above are violated? What parts are satisfied here?