Homework 5
Due Tuesday, 03/11/08

1. Please do problem 1 in Exercise 11.3.4 on page 275 of Bird et al.

   Soln from Matt – cheating and using unify!:

```
import nltk.featstruct

def subsumes(fs1, fs2):
    """ inputs: fs1, a feature structure
                fs2, ditto
        output: True if fs1 subsumes fs2, False otherwise

        Finds the unification of fs1 and fs2, and if that is equal to fs2
        then fs1 subsumes fs2.  Reentrance is considered a 'more specific'
        feature.
    """
    unification = fs1.unify(fs2)
    return (fs1.unify(fs2)) == fs2
```

   Without using unify – from Dave:

```
#May need to check for path equivalency
#Takes two feature structures and returns True if fsOne subsumes fsTwo
def subsumes(fsOne, fsTwo):
    sub = True
    if type(fsOne) == nltk.featstruct.FeatStruct:
        for feat in fsOne:
            if feat in fsTwo:
                sub = sub & subsumes(fsOne[feat],fsTwo[feat])
            else:
                return False
    elif fsOne == fsTwo:
        return True
    return sub
```

2. In this exercise i want you to annotate a grammar in order to make sure that only correct sentences can be generated. You will start with the grammar in file `decGrammar.fcfg`, which contains a grammar to generate simple declarative sentences. It is a simple extension of that shown on page 265 of Bird et al.

   Part a below is a stand-alone exercise. Both parts a and b should build on the original grammar in `decGrammar.fcfg`. However, part c should build on b, and part d on c (though a detailed grammar is not required for d)

   (a) Please expand the grammar to handle the pronouns 'I', 'we', 'you', 'he', 'she', 'they', as well as 'me', 'us', 'him', 'her', and 'them'. this will involve adding new lexical rules (see the starter

grammar) as well as new features to handle the person and whether it appears in the subject or object position. (See problem 16.2 on page 42 of Jurafskey & Martin for examples.)

Test your annotated grammar with commands like the following (after you have imported nltk):

```
>>> cp = load_earley('file:impGrammar.fcfg',trace=2)
>>> tokens = 'I like the dog'.split()
>>> trees = cp.nbest_parse(tokens)
```

Provide test output that shows that your grammar accepts and rejects enough of the appropriate sentences that I can have confidence that it works correctly. (You can use trace=0 on the output for me once you are convinced your grammar works.)

Soln mainly from Erik:

```
% start S
# ############################
# Grammar Rules
# ############################
# S expansion rules
S -> NP[NUM=?n, CASE=nom, PERS=?p] VP[NUM=?n, PERS=?p]
# NP expansion rules
NP[NUM=pl, PERS='3rd'] -> N[NUM=pl]
NP[NUM=?n, PERS='3rd'] -> PropN[NUM=?n]
NP[NUM=?n, PERS='3rd'] -> Det[NUM=?n] N[NUM=?n]
NP[NUM=?n, CASE=?c, PERS=?p] -> Pron[NUM=?n, CASE=?c, PERS=?p]
# VP expansion rules
VP[TENSE=?t, NUM=?n, PERS=?p] -> IV[TENSE=?t, NUM=?n, PERS=?p]
VP[TENSE=?t, NUM=?n, PERS=?p] -> TV[TENSE=?t, NUM=?n, PERS=?p] NP[CASE=acc]
# ############################
# Lexical Rules
# ############################
Det[NUM=sg] -> 'this' | 'every'
Det[NUM=pl] -> 'these' | 'all'
Det -> 'the' | 'some'
PropN[NUM=sg]-> 'Kim' | 'Jody'
N[NUM=sg] -> 'dog' | 'girl' | 'car' | 'child' | 'flight'
N[NUM=pl] -> 'dogs' | 'girls' | 'cars' | 'children'

#Pronouns
Pron[NUM=sg, CASE=nom, PERS='3rd'] -> 'he' | 'she'
Pron[NUM=sg, CASE=nom, PERS='1/2'] -> 'I' | 'you'
Pron[NUM=pl, CASE=nom, PERS='3rd'] -> 'they'
Pron[NUM=pl, CASE=nom, PERS='1/2'] -> 'we' | 'you'
Pron[NUM=sg, CASE=acc, PERS='3rd'] -> 'him' | 'her'
Pron[NUM=sg, CASE=acc, PERS='1/2'] -> 'me' | 'you'
Pron[NUM=pl, CASE=acc, PERS='3rd'] -> 'them'
Pron[NUM=pl, CASE=acc, PERS='1/2'] -> 'us' | 'you'
```

```
#3rd person sing. verbs
IV[TENSE=pres, NUM=sg, PERS='3rd'] -> 'disappears' | 'walks'
TV[TENSE=pres, NUM=sg, PERS='3rd'] -> 'sees' | 'likes' | 'books'

#Verbs for other cases
IV[TENSE=pres, NUM=pl, PERS='3rd'] -> 'disappear' | 'walk'
TV[TENSE=pres, NUM=pl, PERS='3rd'] -> 'see' | 'like' | 'book'
IV[TENSE=pres, NUM=?n, PERS='1/2'] -> 'disappear' | 'walk'
TV[TENSE=pres, NUM=?n, PERS='1/2'] -> 'see' | 'like' | 'book'

#Past verbs
IV[TENSE=past,NUM=?n, PERS=?p] -> 'disappeared' | 'walked'
TV[TENSE=past,NUM=?n, PERS=?p] -> 'saw' | 'liked' | 'booked'
```

(b) Please expand the grammar to handle auxiliary verbs like 'is', 'are', 'do', and 'does'. Your new grammar should recognize sentences like "Jody does like the dog." and "The children do like the dog".

This will involve adding productions like:

```
S -> NP Aux VP
Aux -> 'do' | 'does'
```

although you will also have to add appropriate annotation of features. See the provided grammar for the form for writing the annotated rules. Write your grammar with features in the same fashion. You may add new features to the grammar if necessary.

Make sure your new grammar only accepts grammatical sentences. In particular, you must worry about getting the right forms for the verbs in each case. For example, the following are clearly not right.

- Jody do likes the bike.
- The girls does ride the bike.

Again provide sufficient test data to convince me that your grammar works.

Solution: Add:

```
S -> NP[Num=?n, CASE=nom, PERS=?p] AUX[Num=?n, PERS=?p] VP[TENSE=pres, NUM = pl]
AUX[Num=sg] -> 'does'
AUX[Num=pl] -> 'do'
```

Notice that the auxiliary must match in number with the verb, though the following verb is always in the form associated with non-3rd person singular, which we represent with "pl" (out of laziness!). Note that CASE and PERS tags may be omitted if there are no pronouns.

(c) Please expand the grammar to handle questions and commands. That is, add the rules:

```
S -> Aux NP VP
S -> VP
```

Again provide sufficient test data to convince me that your program works.

Solution: Add:

```
S -> AUX[Num=?n, PERS=?p] NP[Num=?n, CASE=nom, PERS=?p] VP[TENSE=pres, NUM = pl]
S -> VP[TENSE=pres, NUM = pl]
```

(d) Handling "wh-questions" can be challenging because of the distance between words like 'who', 'what', 'where', etc. and the lexical items that they must agree with.

Discuss how you could force agreement for sentences like:

```
What does Jody like?
Who likes the cars?
```

You need not provide detailed grammar rules.