# Homework 3

The assignment is to add features to the grammar and parser, as described in Handout 6. Submit a single file, `hw3.py`.

**1.** Categories.

**a.** Define class `Category` as a subclass of `tuple`. The only method you need to provide is `__repr__()`:

```
>>> from hw3 import *
>>> Category(['V', 0, 'i', '0'])
V.$0.i.0
```

**b.** Define function `parse_category`$(v, t)$ that takes a string $v$ and a symbol table $t$. The symbol table is optional. If the symbol table is not provided, signal an error if $v$ contains variables. An example:

```
>>> t = {}
>>> c = parse_category('A.x.$x', t)
>>> c
A.x.$0
>>> c[2]
0
>>> parse_category('B.$y.$x', t)
B.$1.$0
>>> parse_category('C.x')
C.x
>>> parse_category('C.$x')
Traceback (most recent call last):
  ...
Exception: Variables not allowed
```

**c.** Define `meet`$(u, v)$

```
>>> meet('a', 'a')
'a'
>>> meet('a', 'b')
>>> meet('a', '*')
'a'
>>> meet('*', 'a')
'a'
```

**d.** Define `unify`$(x, y, b)$

```
>>> t = {'x': 0, 'y': 1}
>>> b = ['*','*']
>>> b2 = unify(parse_category('A.$y.b', t), parse_category('A.c.b', t), b)
>>> b2
['*', 'c']
>>> unify(parse_category('B.$y', t), parse_category('B.b', t), b2)
>>>
```

**e.** Define `subst(`$b, x$`)`

```
>>> subst(b2, parse_category('X.$y.b.$x', t))
X.c.b.*
```

**2.** Redefine `Lexicon`, `Rule`, and `Grammar`

**a.** `Index` does not change; you can import it from `hw2`.

**b.** Modify the `load()` method for `Lexicon` so that it accepts lexicons with features.

```
>>> lex = Lexicon('fg0.lex')
>>> lex.parts('barked')
[V.*.i.0]
```

**c.** The constructor is now `Rule(`$x, y, b$`)`, for $x$ the lefthand-side category, $y$ the list of righthand-side categories, and $b$ a list of initial bindings. Add a `bindings` attribute, and make sure `__repr__()` does the right thing.

```
>>> t = {}
>>> np = parse_category('NP.$n', t)
>>> det = parse_category('Det.$n', t)
>>> n = parse_category('N.$n', t)
>>> r = Rule(np, [det,n], ['*'])
>>> r
NP.$0 -> Det.$0 N.$0
```

**d.** Modify the `load()` method for `Grammar`. You may delete the `generate()` method—it will not work with features.

```
>>> g = Grammar('fg0')
>>> g.continuations('Det')
[NP.$0 -> Det.$0 N.$0]
```

**3.** Redefine `Parser` as described in Handout 6. `Node` and `cross_product` do not change; you can import them from `hw2`.

```
>>> p = Parser(g)
>>> trees = p('the dog barks'.split())
>>> print(trees[0])
(S
  (NP.sg
    (Det.* the)
    (N.sg dog))
  (VP.sg
    (V.sg.i.0 barks)))
```

**4.** For a more strenous test, make sure that your parser handles `fg1` and all the sentences in `fg1.sents`.