CS 560: Homework 0

Eric Stevens, October 1, 2018

1. If a language consists only of four 0-nary predicate symbols then it very limited.

    a. The only facts that can be written are the predicates themselves: a(), b(), c() and d().

    b. If all four of the predicates can be contained in a body then the number of possible bodies becomes a summation of combination as follows: $B = \sum_{i=1}^{4} \binom{4}{i} = \sum_{i=1}^{4} \frac{4!}{i!(4-i)!} = \frac{24}{24} + \frac{24}{6} + \frac{24}{4} + \frac{24}{6} = 15$. If you don't allow all four to be used in a body, because you know one must be atom must be used as the head of a rule, then the math only changes slightly: $B = \sum_{i=0}^{3} \binom{4}{i} = \sum_{i=1}^{3} \frac{4!}{i!(4-i)!} = \frac{24}{6} + \frac{24}{4} + \frac{24}{6} = 14$ {a^b^c}, {a^c^d}, {b^d}, {a,b} and {c}.

    c. To calculate the total number of rules that can be created using this set of predicates we use similar math as we used to calculate the number of possible bodies. Here though, there must be a head. Since there can only be one head at a time there are four options for heads. That leaves combinations of the three remaining predicates. The math to calculate the number of possible rules is as follows: $R = 4 \sum_{i=1}^{3} \binom{3}{i} = 4 \sum_{i=1}^{3} \frac{3!}{i!(3-i)!} = 4\left(\frac{6}{6} + \frac{6}{2} + \frac{6}{2}\right) = 28$. Some examples of these rules might be: {d <- a^b^c}, {a <- c^d}, {c <- b^d}, {a <- b} and {a <- c}.

2. Write a datalong that states: *If you have a son then you have a child.*
    a. Child() <- Son()

Question 3

In [1]:

```
list1 = [1, 2, "c", 4, "e"]
list2 = [6, "g", 7, "i", "j"]
```

In [2]:

```
print("list1: ", list1)
print("list2: ", list2)
```

```
list1:  [1, 2, 'c', 4, 'e']
list2:  [6, 'g', 7, 'i', 'j']
```

In [5]:

```
list3 = list1 + list2
print(list3)
```

```
[1, 2, 'c', 4, 'e', 6, 'g', 7, 'i', 'j']
```

In [4]:

```
list4 = [list1,list2]
print(list4)
```

```
[[1, 2, 'c', 4, 'e'], [6, 'g', 7, 'i', 'j']]
```

In [5]:

```
list1.append("x")
print(list1)
```

```
[1, 2, 'c', 4, 'e', 'x']
```

In [6]:

```
list1.extend(list2)
print(list1)
```

```
[1, 2, 'c', 4, 'e', 'x', 6, 'g', 7, 'i', 'j']
```

In [7]:

```
list1 = [1, 2, "c", 4, "e"]
print(list1[0])
print(list1[3])
print(list1[-1])
```

```
1
4
e
```

In [9]:

```
print(list1[1:3])
print(list1[:])
print(list1[2:])
print(list1[-3:-1])
```

```
[2, 'c']
```

```
[1, 2, 'c', 4, 'e']
['c', 4, 'e']
['c', 4]
```

In [10]:

```
print(list1[2:3])
print(list1[2])
```

```
['c']
c
```

In the code above the first line returns a slice of list1 with a range of one index. The second line returns the value at a single index of the list.

In [11]:

```
list1 = [1, 2, "c", 4, "e"]
list2 = list1
list1.append("x")
print(list2)
```

```
[1, 2, 'c', 4, 'e', 'x']
```

In this statement, setting list2 equal to list1 is not actually creating a new list variable and copying data over from list one but rather creating a list pointer object that points to the same object that list one points to.

In [13]:

```
list1 = [1, 2, "c", 4, "e"]
list2 = list1
list1 = list1 + [6]
print(list1)
print(list2)
```

```
[1, 2, 'c', 4, 'e', 6]
[1, 2, 'c', 4, 'e']
```

In this situation you are setting list1 equal to itself and another list. This requires the creation of a new object. List two is still pointing to the first object.

In [6]:

```
list1 = [1, 2, "c", 4, "e"]
list2 = list1[:]
list1.append("x")
print(list1)
print(list2)
```

```
[1, 2, 'c', 4, 'e', 'x']
[1, 2, 'c', 4, 'e']
```

Here we are setting list2 equal to the values within list1 as opposed to list1 itself. In this situation a new object is created and pointed to by list2.

Question 4

In [10]:

```
def parse(atom):
    for x in range(1,len(atom)):
        if atom[x][0].islower():
            print("constant ", atom[x])
        else:
            print("variable ", atom[x])
```

```
            print( "variable" , atom[x])
```

In [11]:

```
myatom = ["pred", "foyer", "X", "foyer", "parlour", "Y", "X"]
parse(myatom)
```

```
constant  foyer
variable  X
constant  foyer
constant  parlour
variable  Y
variable  X
```

Question 5

In [23]:

```
def parse(atom):
    d={}
    for x in range(1,len(atom)):
        if atom[x][0].islower():
            d[atom[x]] = "constant "
        else:
            d[atom[x]] = "variable "

    for i in d:
        print ( d[i], i)
```

In [24]:

```
myatom = ["pred", "foyer", "X", "foyer", "parlour", "Y", "X"]
parse(myatom)
```

```
constant  foyer
variable  X
constant  parlour
variable  Y
```