Cogs 543 Fall 2018
Computational Semantics Assignment 04 due Oct 31, Wednesday

Question 1

A crucial task in model-theoretic semantics is to compute the value of an expression in terms of the objects in a given model. This week we will take another small step toward that goal. For the time being, rather than taking expressions as input, we will take structured representations that organize vocabulary items into an applicative structure. An applicative structure is an hierarchical organization of functions and their arguments. We will construe applicative structures as binary branching trees (represented as nested two membered lists) where for each node the left member will be the function and the right member will be the argument. Here are some examples:

- The sentence *Mary sleeps* will be given as $(sleep'\, mary')$
- The sentence *Tina is happy* will be given as $((is'\, happy')\, tina')$
- The sentence *Tina loves John* will be given as $((loves'\, john')\, tina')$

The primes in the applicative structures are there to indicate that we are not dealing with linguistic items as we read/here them, but some representation more abstract than them.

Given such an applicative structure of an expression, if you know the meanings of its parts, it will be trivial to obtain the meaning of the whole: you simply have to recursively apply each function to its argument. There are a couple of things to be done at this stage. In our random models, vocabulary items that we would want to be interpreted as functions are represented as sets of tuples. As you cannot apply a set to an argument, they need to be transformed into functions. Furthermore, these functions need to be curried. For this week, the task is to write a procedure that turns the set theoretic representations of our vocabulary items into their corresponding functions. In case you cannot figure out what a general function would be, first write functions that do the transformations for 0 1 and 2 place vocabulary items separately, and then try to find a way to generalize.