Cogs 543 Fall 2019
Computational Semantics Take-home due Jan 23

In this take-home you are asked to write a lexicon (and, if needed, a theory) in `SmallWorld`. Please see here for a guide. Best way is to start with the example project and go by modifying it. You may not even need to modify the theory part.

Task 1

The first task is to enrich the semantics of quantifiers currently coded in the example project (`prj/basic`). When you start `SmallWorld` on the example project by typing `main.lisp prj/basic` when inside the `smallworld/code/` directory; you can parse sentences that the example project is capable of. To see the defined vocabulary, type `:sv`.

```
Ready> :p every spy walks

(#S(SIGN
    :PHON ((EVERY . SPY) . WALKS)
    :SYN ((CAT V) (AGR SG) (BAR 1))
    :SEM ((EVERY SPY) WALKS)))
```

The semantics given to the sentence *every spy walks* just gives the applicative form, namely which function should apply to which. The semantics we would want to obtain is

$$\forall x.spy'x \to walks'x$$

The easiest way to encode such logical forms in `SmallWorld` is to treat logical connectives as curried prefix functions. For instance, the formula $spy'x \to walks'x$ would be rendered as

```
((COND (SPY X)) (WALK X))
```

where `COND` becomes a two-place curried function, as in the rest of our semantics.

As for the quantifiers $\forall$ and $\exists$, again the easiest method would be to treat their syntax exactly like lambda. If we do so, the logical form of *every spy walks* would be:

```
Ready> :p every spy walks

(#S(SIGN
    :PHON ((EVERY . SPY) . WALKS)
    :SYN ((CAT V) (AGR SG) (BAR 1))
    :SEM (FORALL X ((COND (SPY X)) (WALKS X)))))
```

The symbols `LAM`, `FORALL` and `EXISTS` are special symbols with the same syntax.

Your task is to modify the example lexicon, so that *every*, *some*, *a* and *no* are correctly defined with their corresponding semantics. Note that the `\$` convention used in lexical entries will not work, so you need to give separate entries for each of the quantifiers.

Task 2

In its current form, the example project can handle only quantified subjects. Write lexical entries that would also work for the object position quantified expressions. For instance, *mary loves every spy* should give $\forall x.spy'x \to loves'xmary'$, while *mary loves some spy* should give $\exists x.spy'x \wedge loves'xmary'$. Their corresponding `SmallWorld` renderings would be `(FORALL X ((COND (SPY X)) ((LOVES X) MARY)))` and (EXISTS X ((AND (SPY X)) ((LOVES X) MARY))).