Homework 10

For this homework, the task is to add semantic translations to a grammar. The following files are provided:

- `hw-10.sents`: A target list of sentences. When you are done, your grammar should interpret them all, and your model should satisfy them all. That is, they should all have the value `True` according to the model.
- `hw-10.fcfg`: An initial grammar. This grammar parses all of the sentences, but it does not compute semantic translations above the word level. To keep things simple, there are no features at all; they are not necessary to handle the given sentences.
- `hw-10.trans`: Target semantic translations for each of the sentences. It initially contains the same dummy translation for every sentence, namely, `REPLACE(ME)`. You will need to replace the dummy translations with the correct translations.
- `hw-10.domain`: The domain. It contains a single line, listing the individuals, namely: `a b f m s`.
- `hw-10.model`: An initial model. It contains names for the individuals, but nothing else. You will need to flesh it out.

The assignment is as follows. `Gdev` has been updated to help you track semantic translations and values: see "Extensions to `gdev`" below.

1. Manually translate each of the sentences to the predicate calculus. Write the translations in the file `hw-10.trans`, replacing the dummy translations that are initially there. For individual words, use the predicate-calculus symbols given in the initial grammar. These translations provide the targets for the translations produced by the grammar. You may end up revising some of these targets as you go; what is important is that the predicate calculus expression correctly reflects the meaning of the sentence.
2. Edit the grammar file `hw-10.fcfg` and add semantic translations to the rules. The only feature you will need to add to the grammar is a feature `v` containing the semantic value of each node. You should not make any changes to the grammar other than the addition of the `v` feature.[1] Your final grammar should correctly translate each sentence to the predicate calculus. That is, the translations according to the grammar should match the target translations you wrote for question 1.
3. Edit `hw-10.model` and flesh out the model. The lines in `hw-10.model` should all have the form *symbol* `=>` *value,* where *value* is either an individual, a set of individuals, or a set of pairs of individuals (representing

[1] Although the book uses `SEM`, you should use `v`. `Gdev` assumes that the semantic translation is the value of `v`.

a two-place relation). For example, if a, b, c, and d are individuals, then the following would be well-formed contents for the model file:

```
MOLLY => a
HOWARD => b
FOO => {a,b,c}
BLIP => {(a,b), (c,d)}
```

When you are done, your model should satisfy all the sentences. That is, each sentence should have the value `True` according to the model. (There is more than one way to write a model that satisfies the sentences; any model that satisfies the sentences is fine.)

A few of the sentences deserve commentary.

- `Max is a red cat`. Note that "a red cat" is a *predicate nominal,* not a QP.
- `every cat is a animal`. For the sake of simplicity, we use "a" instead of "an."
- `Bonnie is not red`. If you have `-RED` as the translation of "not red," then your sentence translation will look right—"`-RED(BONNIE)`"—but it actually has the structure `(-RED)(BONNIE)`, which is incorrect. When you try to evaluate it, you will get the obscure error `'bool' object has no attribute '__getitem__'`.
- `every dog is not a cat`. This sentence is actually ambiguous. You should do the reading in which "every" has wide scope over "not." That is, the sentence asserts that no dog is a cat.

A couple of extra tips:

- Be careful about grouping. In particular, remember that "`.`" has high precedence, which is usually not what you want. Put parentheses around the body of a lambda expression or quantifier: `\x.(RED(x) & DOG(x))`.
- Be sure to do `sems` when you are done, and check that everything is `True`. Later modifications to the model may break things that were working earlier.

Submission. You should submit the three files that you edit: `hw-10.trans`, `hw-10.fcfg`, and `hw-10.model`. You should *not* change the other three files. The files you submit should load correctly in `gdev`.

Extensions to `gdev`

The module `gdev` was described on Handout 20. It has some additional commands that are useful for semantic interpretation. Recall that we start it up by doing:

```
$ python -m gdev hw-10
```

As an alternative, you can start Python and do:

```
>>> from gdev import Grammar
>>> Grammar('hw-10').run()
```

The five files listed at the beginning of the homework should be present in the current directory.

One can get information about the interpretation of the current sentence using the command `sem`:

```
G> sem
[0]
(S[] (NP[] (Name[v=<SPOT>] Spot)) (VP[] (Vi[v=<BARKER>] barks)))
Target:  REPLACE(ME)
Trans: * None
Value:   None
```

This displays the parse tree, the target translation, the translation found at the root of the parse tree, and the value according to the model. The "`*`" after "Trans:" indicates that the translation does not match the target translation. In this case, there is no translation, because the root `S` node has no features. The translation is defined to be the root node's value for the `v` feature.

To change the target translation, edit the file `hw-10.trans`. To get the `S` node to have features, edit the file `hw-10.fcfg`. Once you are producing the correct translation, you will need to edit `hw-10.model` to get the value to be `True`. Each time you make an edit, you will need to do:

```
G> r
G> sem
```

to reload and see the new results. To go on to the next sentence, use the "`n`" command. (See Handout 20.)

One can get a summary of semantic information for all sentences by calling the command:

```
G> sems
[0] 0 * (no translation) (no value)
[1] 0 * (no translation) (no value)
...
```

The print-out contains one line for each parse tree of each sentence. The line shows the sentence number, the parse-tree number, the predicted translation (prefixed with an asterisk, if it does not match the target translation), and the value. When you are done, all the values should be `True`.