Assignment #2: Propositional Semantics

CISC/CMPE 204, Fall 2021
Due by 11:30 PM on Thursday, September 30, 2021

Please read the details and instructions carefully before you begin to work on the problems. The questions in this assignment are relatively straightforward because they are intended to be a practical introduction to propositional semantics.

Because the graders will be processing a large number of files, we require that you use a strict naming convention for your files.

Your computer-readable answers must be in a single Python script named **a2.py** The assignment statement in onQ will direct you to download an individualized problem set. Your set may use the variable names P, Q, R, S, and T. Your answer must use all and only *your individualized* variables.

There are a total of six statements, or well-formed formulae, in your problem set for this assignment. These statements are defined as logical formulae in variables named s1–s6. The first four statements, s1–s4, are for use in Question 1. The last two statements, s5–s6, are for use in Question 2.

Question 1: Equivalence and Satisfiability

For each statement s1–s4, you are to:

1.1: Write the truth table for the statement (*optional and unmarked*)

1.2: Determine equivalence of statements by comparing their respective truth tables (**1pt**)

1.3: State a model that is satisfied by one group of statements and that is not satisfied by any of the other statements (**1pt**)

For this question, you need to create a folder that contains four files; you will put your computer-readable answers in your Python script named **a2.py**. The folder will contain all of your answers to Question 1.1. The script will contain your answers to Question 1.2, Question 1.3, and Question 2.

The answers to Question 1.1 will be files that are named a2q1a–a2q1d. These can be text files, pictures of hand-written answers, or any other format. They are optional, but will be extremely helpful to solving 1.2 and 1.3.

The answers to Question 1.2 and Question 1.3 will be in your Python script. The script will include the definition of two variables, a2q12 and a2q13. For Question 1.2, statement equivalence, the variable a2q12 will be a list of lists that group together the equivalent statements. These examples are incorrect but illustrate how to answer:

```
#a2q12 = [ [s1,s2,s3,s4] ] # they are all equivalent.
#a2q12 = [ [s1], [s2], [s3], [s4] ] # they are all different
#a2q12 = ??? # the right answer please!
```

For Question 1.3, statement satisfiability, your answer will be a dictionary mapping variables to True or False. Use all and only the variables in your individualized problem set. An example answer, which is almost certainly incorrect, is

```
a2q13 = {
        P: True,
        Q: False,
        R: True,
        S: False,
        T: True
}
```

Question 2: Normal Forms

For each statement s5–s6, we recommend that you draw or write the parse tree for each statement. This may clarify the structure of each statement and assist you in converting each statement to a normal form.

If you wish, you can include your parse trees as files named a2q2s5 and a2q2s6. You can do this on paper and take a photo, or use drawing software. This will not be graded, and TAs will only look at it if you request (e.g., if you want us to confirm you have it done right). While ungraded, they may be instrumental in aiding with questions 2.1-2.3 and the quiz will ask a similar question that *will* be graded.

For each relevant statement, you are to:

2.1: Convert the statement to negation normal form (**0.5pt x 2**)

2.2: Convert the statement to conjunctive normal form (CNF) by using distribution and de Morgan's Rules (**0.5pt x 2**)

2.3: Encode the statement with the Tseitin encoding (**0.5pt x 2**)

Your explanation of each step in the conversion for 2.1 and 2.2 is for the grader to understand how you converted each statement. Example explanation text might be:

- starting formula
- de Morgan
- distribution
- replace implications

- double negation

Your answers to this question will be the definition of four variables in your Python script. For example, suppose that your individualized statements were defined as

```
s5 = ~ (P | Q)
s6 = (P & Q) | R
```

Your answers will be the definition of the four variables a2s5nnf, a2s6nnf, a2s5cnf, and a2s6cnf. For the above statements, example answers might be…

```
a2s5nnf = [
        [˜(P | Q), ’starting formula’],
        [˜P & ˜Q, ’de Morgans’]
]

a2s6nnf = [ [(P & Q) | R, ’starting formula -- already in negation normal form’] ]

a2s5cnf = [
        [˜(P | Q), ’starting formula’],
        [˜P & ˜Q, ’de Morgans’]
]

a2s6cnf = [
        [(P & Q) | R, ’starting formula’],
        [(P | R) & (Q | R), ’distribution’]
]
```

For question 2.3, you are to define each of the steps in a Tseitin encoding for both s5 and s6. Using the examples above, the code would look like:

```
a2s5tseitin = semantic_interface.Encoding()
# first argument is the formula; second is the variable name.
x1 = a2s5tseitin.tseitin(P | Q, 'x1')
x2 = a2s5tseitin.tseitin(~x1, 'x2')
# This final step is required -- use your last variable, corresponding to the top
#  of the parse tree, to finalize your Tseitin encoding.
a2s5tseitin.finalize(x2)

a2s6tseitin = semantic_interface.Encoding()
# first argument is the formula; second is the variable name.
x1 = a2s6tseitin.tseitin(P & Q, 'x1')
x2 = a2s6tseitin.tseitin(x1 | R, 'x2')
# This final step is required -- use your last variable, corresponding to the top
#  of the parse tree, to finalize your Tseitin encoding.
a2s6tseitin.finalize(x2)
```

Question 3: Your Creation

If you had control of building a quiz to assess knowledge of what the students in 204 learned for propositional semantics,what question would you ask? Keep in mind that there are roughly 3-4 questions / quiz, and only 50min in total.

The only constraint is that you shouldn't just mirror one of the assignment or weekly practice questions. Try to be creative, and the best questions very well may appear on a future quiz!

Leave your answer in a file named **a2q3.txt**. This question is worth **1pt**.

What to turn in

- You will submit your answers electronically as one file that is named a2 xxxxxxxx.zip, where xxxxxxxx is your Queen's University student number. This is essential for the TAs to mark your assignment in a reasonable time.

- Your ZIP file must contain only the files for your answers to this assignment:
  - a2q1a–a2q1d, truth tables
  - a2.py, solution variables
  - a2p3.txt, proposed question
  - config.py, your configuration file
  - *optional*: a2q2s5 and a2q2s6, parse trees

- The code will be tested by one or more graders. The commentary in your answers for Question 2 will be read by one or more graders and will be checked.

- The assignment must be submitted using the Queen's "onQ" software.

Policies

- You must complete these questions individually.

- Although you are allowed to discuss the questions with other students, you must write your own answers and Python code.

- Please refer to the syllabus for details on submitting after the due date.

Statement of Academic Integrity