UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

CSE 15: Discrete Mathematics
Laboratory 2

Spring 2020

Introduction

This lab gives you further practice with the Python language on problems related to truth tables in propositional logic. Please complete the following exercises and upload your `.py` files under the appropriate CatCourses assignments.

Even for the simplest programs we write, we make use of code written by others. For this lab you are provided with code that generates truth tables, which you can use for other exercises. To start, download the file `logic.py` from CatCourses and place it in your working directory. Among other things, the file defines a `TruthTable` object. Start a new project and import the `TruthTable` object from `logic.py`, using the following command:

```
from logic import TruthTable
```

You can now generate truth tables for any proposition. All you need to specify is the list of variables that appear in your propositions, and the propositions themselves.

Example Generate a truth table for the proposition $p \vee q$.

Solution We first note that there are two variables, p and q, which will be represented in Python as the list `['p', 'q']`. We also need to represent the proposition itself, which is the Python string `'p or q'`. Since the `TruthTable` object can generate a truth table with multiple expressions, we will provide the proposition as a list with a single element in it, `['p or q']`. We are now ready to generate the truth table:

```
myTable = TruthTable(['p', 'q'], ['p or q'])
```

We can now display the truth table generated above:

```
myTable.display()
```

The command above produces the following text:

```
p       q       p or q
----------------------
0       0       0
0       1       1
1       0       1
1       1       1
```

You can also generate LaTeX code for representing the table:

```
myTable.latex()
```

The above command produces the following text:

```
\begin{tabular}{|c|c|c|}
\hline
$p$ & $q$ & $p \lor q$ \\
\hline
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\hline
\end{tabular}
```

The `TruthTable` object can also be called with multiple propositions.

```
myTable = TruthTable(['p', 'q'], ['p or q', 'p and q'])
```

The command above generates one truth table with both propositions side by side:

```
p       q       p or q   p and q
--------------------------------
0       0       0        0
0       1       1        0
1       0       1        0
1       1       1        1
```

When using `TruthTable`, we need to specify all propositions and propositional variables as Python strings, as illustrated in the examples here. The `TruthTable` object supports the following logical connectives:

`or` ($\vee$), `and` ($\wedge$), `-` ($\neg$), `->` ($\rightarrow$), `<->` ($\leftrightarrow$).

Basic Truth Tables

Write a Python program that prints out the truth tables for all logical connectives studied in class. There should be a separate truth table for the following:

$\neg a$
$a \wedge b$
$a \vee b$
$a \rightarrow b$
$a \leftrightarrow b$

Equivalence in Propositional Logic

As discussed in class, one way to prove whether two propositions are equivalent, is to generate truth tables for both propositions and check to see if all the rows match, in which case the propositions are equivalent, otherwise they are not.

In this exercise you are required to write a Python program that asks the user to enter two propositions, where both propositions the variables `p` and `q`. Your program should then determine whether or not the two propositions are equivalent, and report the result.

Here is an example of how a user might interact with your program:

```
Enter proposition 1: p and q
Enter proposition 2: q and p

The propositions are equivalent
```

In order to complete this exercise, you should study the structure of the `table` property of `TruthTable`. You can access it by saying:

```
print myTable.table
```

For example, here is the `table` of the proposition $p \wedge q$.

```
[[[0, 0], [0]], [[0, 1], [0]], [[1, 0], [0]], [[1, 1], [1]]]
```

As you can see, it is one big list, made up of exactly 4 smaller lists. Each one of the smaller lists represents a row of the truth table. Each row in turn is made up of exactly 2 lists. The first one is a list of truth values of the propositional variables, and the second one is the overall truth value of the proposition(s), given the truth values in the last list.

If we take the above `table` and consider the first row, that is the list `[[0, 0], [0]]`. Bearing in mind that this is for the formula $p \wedge q$, the way to interpret this list is that when the variables p and q are `[0, 0]`,

respectively, then the proposition $p \wedge q$ has a truth value of `[0]`, or simply `0`. Remember, this is represented as a list with a single element in it because the truth table may have more than one proposition being evaluated.

This brings us to the following example. Let us generate a single truth table for both $p \wedge q$ and $q \wedge p$. Now we have the following `table`:

```
[[[0, 0], [0, 0]], [[0, 1], [0, 0]], [[1, 0], [0, 0]], [[1, 1], [1, 1]]]
```

It is still made up of 4 lists, each of which represents a row of the truth table. Each row is still made up of two lists, as before. The first of these lists is still a specific combination of truth values, and the second list is the overall truth value of each proposition. Since we generated a truth table for 2 propositions, the list has 2 results in it. This time, taking the second row, which is `[[0, 1], [0, 0]]`, we can see that when $p = 0$ and $q = 1$ then $p \wedge q$ evaluates to false, and $q \wedge p$ also evaluates to false, hence both `0` values in the list. Examining the whole table, we can see that the two propositions are indeed equivalent because, for every row, the result part always contains the same two values, meaning that for any possible combination of truth values, both propositions evaluate to the same result.