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

Fall 2020

- This lab is meant to give you some practice with the Python programming language. Please complete the following exercises and upload your `.py` files under the appropriate CatCourses assignments.
- **This lab assignment will be graded and its grade will be counted towards computing the course grade**.
- **This lab must be solved individually.** You can discuss your ideas with others, but when you prepare your solution you must work individually. Your submission must be yours and yours only. No exceptions, and be reminded that the CSE academic honesty policy discussed in class will be enforced.
- Your solution must be exclusively submitted via CatCourses. Pay attention to the posted deadline because the system automatically stops accepting submissions when it expires.
- Start early.

Introduction

In this assignment 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` ($\wedge$), `-` ($\neg$), `->` ($\rightarrow$), `<->` ($\leftrightarrow$).
Note that when evaluating compound propositions with multiple operators `TruthTable` applies the precedence rules we saw in class. As always, if you are in doubt, use parentheses.

Warm Up

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:

$a \wedge \neg b$
$(a \wedge b) \vee \neg c$

Rules of Inference

Using the `TruthTable` object, write a program that verifies the tautologies associated with all the rules of inference we saw in class (repeated in the following for your convenience)

Rule	Tautology	Name
$p \rightarrow q$ p $\therefore q$	$((p \rightarrow q) \wedge p) \rightarrow q$	Modus Ponens
$\neg q$ $p \rightarrow q$ $\therefore \neg p$	$(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$	Modus Tollens
$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$	$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$	Hypothetical Syllogism
$p \vee q$ $\neg p$ $\therefore q$	$((p \vee q) \wedge \neg p) \rightarrow q$	Disjunctive Syllogism
p $\therefore p \vee q$	$p \rightarrow (p \vee q)$	Addition
$p \wedge q$ $\therefore p$	$(p \wedge q) \rightarrow p$	Simplification
p q $\therefore p \wedge q$	$((p) \wedge (q)) \rightarrow (p \wedge q)$	Conjunction
$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	$((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$	Resolution