Discrete Structures

Worksheet 1

Spring 2021
January 6/8

1. **Warm up:** Answer the following questions, where P is a proposition.
   (a) True or False: If P is unsatisfiable, then P is a contradiction.
   (b) True or False: If P is satisfiable, then P is a tautology.
   (c) What are de Morgan's laws?

2. Complete the following truth table.

P	Q	$(P \to Q) \leftrightarrow (\neg Q \to \neg P)$	$(P \lor Q) \lor (\neg P \lor Q)$	$(P \land \neg P) \lor (Q \land \neg Q)$
T	T			
T	F			
F	T			
F	F			

3. Consider the following truth table, where S_1, S_2, S_3 are logical propositions.

P	Q	S_1	S_2	S_3
T	T	T	F	T
T	F	T	T	F
F	T	F	T	T
F	F	F	T	F

   (a) Construct two propositions with different symbols that are logically equivalent to each of S_1, S_2, S_3, using only logical quantifiers and P and Q.
   (b) Is it possible to define a proposition logicall equivalent to S_3 using only logical quantifiers and S_1 and S_2? Why or why not?

4. Let P and Q be logical propositions. The following propositions are being simplified to show they are contradictions. Name the law or definition being used in each line.

proposition	law applied
$\neg((P \lor \neg P) \lor T)$	*given*
$\neg(P \lor (\neg P \lor T))$	
$\neg(P \lor \neg P)$	
$(\neg P) \land (\neg(\neg P))$	
$\neg P \land P$	
F	

proposition	law applied
$\neg(\neg(\neg(T \lor T) \land \neg(P \land \neg P)))$	*given*
$\neg(\neg(\neg(T \lor T)) \lor \neg(\neg(P \land \neg P)))$	
$\neg((T \lor T) \lor (P \land \neg P))$	
$\neg(T \lor (P \land \neg P))$	
$\neg(T \lor F)$	
$\neg T$	
F	

5. Rewrite the following propositions using only symbols.

   (a) Every natural number is a rational number.

   (b) Not every real number is a natural number.

   (c) Dividing a real number by a rational number always gives a positive or negative natural number.

6. Consider the five propositions below.

   (P_1) I like pineapple on my pizza.
   (P_2) All odd-numbered propositions are false.
   (P_3) All even-numbered propositions are true.
   (P_4) At least one of P_2 or P_3 is true.
   (P_5) If P_1 is false then P_2 is true.

   Some of these propositions refer to other propositions on the list. Notice that if P_3 is true then all even-numbered propositions must be true, and so P_2 must be true. The truth of P_2 implies all odd-numbered propositions are false, and so P_3 is false. So if P_3 is true then it must also be false. This contradiction means P_3 must not be true.

   Assign the truth values *True* and *False* to each of the above five propositions so that there are no contradictions, or say why it is not possible.

7. There are 100 propositions written on a piece of paper, as below:

   (1) Exactly 1 of the propositions on this paper is false.
   (2) Exactly 2 of the propositions on this paper are false.
   (3) Exactly 3 of the propositions on this paper are false.
   ...
   (100) Exactly 100 of the propositions on this paper are false.

   Which of these propositions are true, and which ones are false?

The next three questions are to be completed using the programming language *Python*. This, and other languages, will be used from time to time in this course.

8. (a) Log into `colab.research.google.com` with your RBS credentials and open up a new notebook.

   (b) In the notebook, type the following text in a cell and press `Shift` and `Enter` at the same time (this is called *execution*). What is the result?

```
True or False
```

   (c) Type each of the following lines in a new cell and execute the cell. What is the result for each?

Text	Result
`True or False`	
`True and False`	
`True or False and False`	
`False or True and False`	
`(True or False) and False`	

(d) Place sets of parenthesis `( )` in the following proposition in two different ways so that the result is `True`, and two different ways so that the result is `False`.

```
True or False and False or False
```

9. (a) Type the following code in one cell in Colab and execute it:

```
def addormultiply(number):
    L = [10, 15, -7, 2.2]
    for n in L:
        largest = max(n+number, n*number)
        print(largest)
```

(b) In the next cell, execute `addormultiply(1)` and `addormultiply(2)`.

(c) Change the values of the four elements of `L` in the definition of `addormultiply` so that they are still different, but so that `addormultiply(3)` shows only two different numbers.

(d) Is it possible to change the values of the four elements of `L` in the definition of `addormultiply` so that `addormultiply(1)` and `addormultiply(2)` both print the same numbers? If yes, what should the values be? If no, why not?

10. Type in the following code in a new cell and execute it.

```
import random
random.choices(['T','F'],k=8)
```

Enter the results into the fourth column of the table below, viewing them as evaluations of a proposition S involving three other statements P, Q, R.

P	Q	R	S (random values)	CNF of S
T	T	T		
T	T	F		
T	F	T		
T	F	F		
F	T	T		
F	T	F		
F	F	T		
F	F	F		

Complete the fifth column, writing a possible conjunctive normal form (CNF) for each proposition S with the given values of P,Q,R so that S evaluates to the given entry in the fourth column.

Appendix 1: Create the DNF and CNF

Each Boolean function $f(P_1, P_2, \ldots, P_n)$ of n propositional variables can be represented as a full Disjunctive Normal Form (DNF) and a full Conjunctive Normal Form (CNF). And this representation is unique – there is exactly one full DNF and exactly one full CNF (up to the ordering of variables and subexpressions).

Below is an example $n = 3$, and there are three propositional variables A, B and C.

A	B	C	$f(A,B,C)$	DNF	CNF
F	F	F	F		$(A \vee B \vee C) \wedge$
F	F	T	T	$(\neg A \wedge \neg B \wedge C) \vee$	
F	T	F	T	$\vee (\neg A \wedge B \wedge \neg C) \vee$	
F	T	T	T	$\vee (\neg A \wedge B \wedge C) \vee$	
T	F	F	T	$\vee (A \wedge \neg B \wedge \neg C) \vee$	
T	F	T	F		$\wedge (\neg A \vee B \vee \neg C) \wedge$
T	T	F	F		$\wedge (\neg A \vee \neg B \vee C)$
T	T	T	T	$\vee (A \wedge B \wedge C)$	

Here is the full DNF written as a disjunction of 5 terms (each term is a conjunction of the variables or their negations):

$$f(A,B,C) = (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge B \wedge C).$$

And here is the full CNF – a conjunction of 3 terms (each term is a disjunction):

$$f(A,B,C) = (A \vee B \vee C) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C).$$

Full DNF is unique, but sometimes there exist other DNFs that are not full (not every term contains all n variables). For example, one can combine Lines 3 and 4 in the above truth table like this:

$$(\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \;\equiv\; (\neg A \wedge B) \wedge (C \vee \neg C) \;\equiv\; (\neg A \wedge B).$$

Therefore we can simplify the full DNF written above:

$$f(A,B,C) = (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge B \wedge C) \equiv$$

$$\equiv (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge B \wedge C).$$

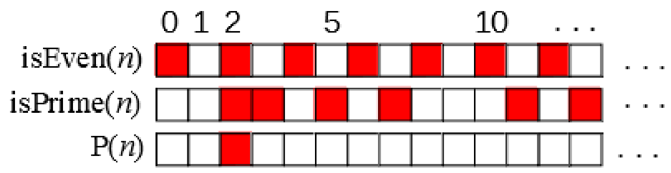
This is also called DNF (a disjunction of conjunctions), albeit it is not a full one.

Appendix 2: Introduction to Predicates and Quantifiers

Let $\mathbf{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers, and we define some functions from $\mathbf{N}$ to the Boolean values $\{\mathtt{F}, \mathtt{F}\}$. Such functions will be called *predicates*.

$$\begin{cases} \text{isEven}(n) := n \text{ is even} \\ \text{isPrime}(n) := n \text{ is a prime number} \\ P(n) := \text{isEven}(n) \wedge \text{isPrime}(n) \end{cases}$$

If we show every true value as a red square and every false value as a white square, we get the following picture:



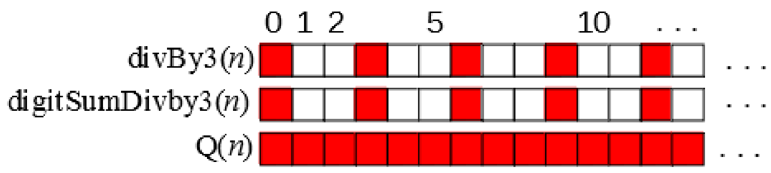
The bottom row displays a red square iff the first two rows had a red square (that is how conjunction $\wedge$ works). We can see that the bottom row has a red square. This can be written as a short formula:

$$\exists n \in \mathbf{N},\ P(n) \quad \equiv \quad \exists n \in \mathbf{N},\ (\text{isEven}(n) \wedge \text{isPrime}(n))\,. \tag{1}$$

The symbol $\exists$ is called *existential quantifier*. It asserts that there is at least one (maybe, more) values $n \in \mathbf{N}$ satisfying our Boolean expression. Formula (1) is pronounced like this: *There exists a natural number n, such that n is even and n is prime.*

Similarly, we define another set of predicates:

$$\begin{cases} \text{divBy3}(n) := n \text{ is divisible by } 3 \\ \text{digitSumDivBy3}(n) := \text{the sum of digits for the number } n \text{ is divisible by } 3 \\ Q(n) := \text{divBy3}(n) \leftrightarrow \text{digitSumDivBy3}(n) \end{cases}$$



The bottom row displays all squares as red (because the above two rows have the same places with values true and false, so they always satisfy the biconditional $\leftrightarrow$). This can be written as a short formula:

$$\forall n \in \mathbf{N},\ Q(n) \quad \equiv \quad \forall n \in \mathbf{N},\ (\text{divBy3}(n) \leftrightarrow \text{digitSumDivBy3}(n))\,. \tag{2}$$

The symbol $\forall$ is called *universal quantifier*. It asserts that the statement is true for every element n of the domain set $\mathbf{N}$. Formula (2) is pronounced like this: *For any natural number n, n is divisible by 3 iff the sum of its digits is divisible by 3.*