CAB203
Tutorial 3

1 Recursion

1. For each of the following recursive definitions, determine how many bases cases there are and evaluate $f(4)$.

   (a)

$$f(n) = \begin{cases} 0 & : n = 0 \\ 2 - f(n-1) & : n \geq 1 \end{cases}$$

Solution: There is one base case.

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 2 - 0 = 2 \\ f(2) &= 2 - 2 = 0 \\ f(3) &= 2 - 0 = 2 \\ f(4) &= 2 - 2 = 0 \end{aligned}$$

   (b)

$$f(n) = \begin{cases} 1 & : n = 0 \\ 1 & : n = 1 \\ f(n-1) - f(n-2) & : n \geq 2 \end{cases}$$

Solution: There are two base cases.

$$\begin{aligned} f(0) &= 1 \\ f(1) &= 1 \\ f(2) &= 1 - 1 = 0 \\ f(3) &= 0 - 1 = -1 \\ f(4) &= -1 - 0 = -1 \end{aligned}$$

   (c)

$$f(n) = \begin{cases} 0 & : n = 0 \\ 1 & : n = 1 \\ 1 & : n = 2 \\ f(n-1) - f(n-2) + 2f(n-3) & : n \geq 3 \end{cases}$$

Solution: There are three base cases.

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(2) &= 1 \\ f(3) &= 1 - 1 - 2(0) = 0 \\ f(4) &= 0 - 1 + 2(1) = 1 \end{aligned}$$

2. *(Stretch question)* In the lecture we discussed the following recursive definition for expressions.

$$\begin{aligned} EXPR &:= \begin{cases} VALUE \\ EXPR \text{ ``}+\text{''} \ VALUE \\ EXPR \text{ ``}-\text{''} \ VALUE \end{cases} \\ VALUE &:= \begin{cases} CONSTANT \\ VARIABLE \end{cases} \end{aligned}$$

Extend the definition to allow *, /, (and) properly.

Solution:

$$\begin{aligned} EXPR &:= \begin{cases} VALUE \\ EXPR \quad OPERATOR \quad VALUE \\ \text{``(''} \quad EXPR \quad \text{``)''} \end{cases} \\ OPERATOR &:= \begin{cases} \text{``}+\text{''} \\ \text{``}-\text{''} \\ \text{``}*\text{''} \\ \text{``}/\text{''} \end{cases} \\ VALUE &:= \begin{cases} CONSTANT \\ VARIABLE \end{cases} \end{aligned}$$

Propositions

1. Which of the following are propositions? Which are atomic propositions? Which are compound propositions?

   (a) "This sentence is true if and only if it is false."

   Solution: Not a proposition because it does not have a well defined truth value.

   (b) "Please open your exam booklet and begin."

   Solution: Not a proposition because it does not have a truth value.

(c) "Goats eat frogs and paper."

> **Solution:** Compound proposition. It can be divided into "Goats eat frogs." and "Goats eat paper."

(d) "The average human eats five spiders per day."

> **Solution:** Atomic proposition. It is not possible to divide it into smaller propositions.

(e) "It is going to rain today."

> **Solution:** Atomic proposition.

(f) "Socrates is a human if and only if trees are animals."

> **Solution:** Compound proposition. It contains the propositions "Socrates is a human" and "trees are animals."

Logical operators

1. Determine whether each compound proposition is true:

(a) $(1 + 1 = 2) \wedge T$.

> **Solution:** True. The first proposition is true, and so is T.

(b) Socrates is human $\vee$ tomatoes are blue.

> **Solution:** True. The first proposition is true, so at least one of the two is true.

(c) Socrates is human $\oplus$ tomatoes are red.

> **Solution:** False. Both propositions are true, since XOR is exclusive, the overall proposition is false. ($T \oplus T = F$)

(d) Tomatoes are red $\rightarrow$ Socrates is a teapot.

> **Solution:** The first proposition is true, and the second is false. So the whole thing is false ($T \rightarrow F = F$)

(e) $\neg$ Tomatoes are red.

> **Solution:** False. Since Tomatoes are red, the NOT of this statement is false.

(f) Socrates is a teapot $\rightarrow$ $(1 + 1 = 2)$.

Solution: True. The left proposition is false, so the implication is true ($F \to T = T$).

(g) Socrates is a teapot $\leftrightarrow$ Tomatoes are blue.

Solution: True. Both propositions are false, and for $\leftrightarrow$ this means that the whole statement is true ($F \leftrightarrow F = T$).

Formulas

1. Which of these are well-formed formulas?

(a) $\neg\neg p$

Solution: Yes. $\neg p$ is a formula, and so is $\neg\neg p$

(b) $(p \vee q \to)s$

Solution: No, this is not well formed. In particular, $\to$ needs to have another formula to its right, rather than the closing).

(c) $(p \wedge q) \vee p$

Solution: Yes, this is well formed. $(p \wedge q)$ is well formed, and we can OR this with p.

(d) $\neg p \to (pq)$

Solution: No, not well formed since (pq) is not well formed.

2. Use a truth table to determine whether each of these formulas is contingent, a tautology, or a contradiction

(a) $(\neg A \wedge B) \vee A$

Solution:

A	B	$\neg A$	$\neg A \wedge B$	$(\neg A \wedge B) \vee A$
T	T	F	F	T
T	F	F	F	T
F	T	T	T	T
F	F	T	F	F

The last column contains both T and F, so this is contingent.

(b) $(A \wedge B) \to \neg A$

Solution:

A	B	$\neg A$	$A \wedge B$	$(A \wedge B) \to \neg A$
T	T	F	T	F
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

The last column contains both T and F, so this is contingent.

(c) $(A \wedge B) \wedge \neg B$

Solution:

A	B	$\neg B$	$A \wedge B$	$(A \wedge B) \wedge \neg B$
T	T	F	T	F
T	F	T	F	F
F	T	F	F	F
F	F	T	F	F

The last column contains only F, so this is a contradiction.

(d) $(A \wedge A) \vee \neg A$

Solution:

A	$\neg A$	$A \wedge A$	$(A \wedge A) \vee \neg A$
T	F	T	T
F	T	F	T

The last column contains only T, so this is a tautology.

Logical equivalence

1. Use a truth table to show $A \to B \equiv \neg A \vee B$

Solution: We construct the table:

A	B	$\neg A$	$A \to B$	$\neg A \vee B$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Now we look at the columns for $A \to B$ and $\neg A \vee B$ and notice that they are the same.

2. Use substitutions and the logical equivalences from the slides to show $\neg A \to T \equiv T$.

Solution:

$$\begin{aligned} \neg A \to T &\equiv \neg\neg A \vee T \\ &= A \vee T \\ &= T \end{aligned}$$

The first step uses $P \to Q \equiv \neg P \vee Q$. From there, we have a double negative: $\neg\neg A \equiv A$. Finally, we have $A \vee T \equiv T$. Chaining them all together we get the desired equivalence.

3. *(Stretch question)* To start with, we define the NAND operation $\overline{\wedge}$ to be AND followed by a NOT:

$$A \,\overline{\wedge}\, B \equiv \neg(A \wedge B)$$

Please note that $\barwedge$ is *not* standard notation. NAND is *functionally complete* meaning that you can rewrite *any* Boolean formula in just NANDs and parentheses. To see this, show that the following can all be rewritten with just NANDs and paretheses: $A \wedge B$, $A \vee B$, $\neg A$,and $A \rightarrow B$. As an example:

$$A \wedge B \equiv (A \barwedge B) \barwedge (A \barwedge B)$$

Solution:

$$\begin{aligned} A \wedge B &\equiv (A \barwedge B) \barwedge (A \barwedge B) \\ A \vee B &\equiv (A \barwedge A) \barwedge (B \barwedge B) \\ \neg A &\equiv A \barwedge A \\ A \rightarrow B &\equiv A \barwedge (B \barwedge B) \end{aligned}$$

4. *(Stretch question)* Choose some of the logical equivalences listed in lecture 4 and show that they hold using truth tables.

Solution: Depends on what you pick! They are all straightforward applications of truth tables.

5. *(Stretch question)* Explain why $A \equiv B$ is the same thing as saying $A \leftrightarrow B$ is a tautology.

Solution: $A \equiv B$ means that the A and B columns in a truth table always have the same truth value. But then $A \leftrightarrow B$ will always be true, since it is true when A and B have the same truth value. This is just the same as saying that $A \leftrightarrow B$ is a tautology.

2 Python and logic

1. Write a Python function that takes three arguments, x, y, z and returns the value of the formula $(x \rightarrow y) \wedge z$

Solution: We don't have $\rightarrow$ in Python, so substitute with a logical equivalence to get

$$((\neg x) \vee y) \wedge z$$

then implement:

```
def f(x,y,z):
   return ((not x) or y ) and z
```

2. Write a function in Python that takes function $f(x, y)$ (which implements a Boolean formula in two variables), and prints out whether f is a tautology, contradiction, satisfiable or contingent (note that f may be more than one these.)

Solution: We need to test all four possible combinations of T/F for x and y and check the behaviour of $f(x, y)$ in each case. There are lots of ways of doing this. This solution uses loops, which is easier to generalise to more variables.

```
def classify(f):
   tautology = True
   contradiction = True
   for x in {True, False}:
      for y in {True, False}:
         if f(x,y):
            # found a true outcome, can't be contradiction
            contradiction = False
         else:
            # found a false outcome, can't be tautology
            tautology = False
   if contradiction:
      print('Contradiction')
   elif tautology:
      print('Tautology')
      print('Satisfiable')
   else:
      print('Contingent')
      print('Satisfiable')
```

3. Write a function in Python that takes two functions, $f(x, y)$ and $g(x, y)$ (which both implement Boolean formulas in two variables) and prints out whether they are logically equivalent or not.

Solution: As in the previous question, we need to test all four possible combinations of T/F for x and y, and check whether $f(x, y) = g(x, y)$.

```
def testLE(f,g):
   for x in {True, False}:
      for y in {True, False}:
         if f(x,y) != g(x,y):
            print('Not logically equivalent')
            return
   print('Logically equivalent')
```