MATH230: Tutorial One

Propositional Logic: Some Antics

Key ideas

- Identify the propositional structure of an argument,
- Translate natural language to propositional logic,
- Write truth tables,
- Determine whether an argument is semantically valid,
- Use valuations to determine validity.

Relevant lectures: Lectures 1,2, and 3
Relevant reading: L∃∀N Chapters 1,2,6

Hand in exercises: 1b,3b,4,5a,6,8c
Due following Friday @ 5pm to the tutor, or lecturer.

Discussion Questions

1. Translate the following English argument into the formal language for propositional logic. Clearly state the atomic propositions, hypotheses, and the conclusion of the argument.

   I will either go to Matukituki or Rakiura. If I go to Matukituki, then I will go hiking. If I go to Rakiura, then I will go hiking. Therefore, I will go hiking.

2. Show $\{A \vee B, A \to C, B \to C\} \models C$.

3. Show $\models A \vee \neg A$.

Tutorial Exercises

1. Translate the following English arguments into the formal language for propositional logic. Clearly state the atomic propositions, hypotheses, and the conclusion of the argument.

   (a) Moriarty knows Irene is either at work, or at home. He has heard from others that she is not at home. Therefore, he concludes she must be at work.

   (b) If Lestrade observes, then he will solve the crime. If Lestrade does not observe, then he calls for Holmes. As ever, Lestrade sees, but does not observe. Therefore he must call Holmes.

   (c) If Robert rushes, then he will blunder his queen. If Robert does not rush, then he will blunder his queen. Therefore, Robert will blunder his queen.

   (d) Either the vicar is a liar (L), or he shot the earl (V). For, either the vicar shot the earl or the butler did (B). And unless the vicar is a liar, the butler was drunk at nine o'clock (D). And if the butler shot the earl, then the butler wasn't drunk at nine o'clock.

   (e) We will win, for if they attack if we advance, then we will win, and we won't advance.

2. Make a truth table for each of the following statements.

   (a) $P \wedge \neg Q$

   (b) $(R \vee S) \wedge \neg R$

   (c) $(A \vee B) \wedge (A \vee C)$

   (d) $X \to \neg Y$

   (e) $(P \to R) \vee (Q \leftrightarrow S)$

3. **Logical Fallacies** Write truth tables for each of the following arguments. Identify a counterexample in each truth table. What does this say about the validity of each argument?

   (a) $A \vee B$, A. Therefore, $\neg B$.

   (b) $P \to Q$, Q. Therefore, P

   (c) $P \to Q$. Therefore, $\neg P \to \neg Q$.

4. **Truth Valuations**

   Valuations are functions which take in propositional formulae and return 0 or 1 according to whether the formula is true or false. Assigning valuations to the atomic propositions in the formulae is enough to determine the valuation of compound formulae.

   For example, if we know the truth value $v(P)$ of a proposition P, then we can calculate the truth value of the negation $v(\neg P) = 1 - v(P)$.

   Determine similar arithmetic formulae for computing the truth valuations of compound formulae consisting of $\neg, \vee, \wedge, \to$, and $\leftrightarrow$.

5. Verify the following claims of semantic consequence.

   (a) $A \models \neg\neg A$
   (b) $(A \wedge B) \to C \models A \to (B \to C)$
   (c) $A \to B \models A \to (A \wedge B)$
   (d) $A \to B, A \to \neg B \models \neg A$

6. **Principle of Explosion** Use a truth table to show $P \wedge \neg P \models Q$

7. Determine whether the following are tautologies, satisfiable, or contradictions.

   (a) $P \vee (\neg P \wedge Q)$
   (b) $(X \vee Y) \leftrightarrow (\neg X \to Y)$
   (c) $(A \wedge \neg B) \wedge (\neg A \vee B)$
   (d) $\neg(A \to A)$
   (e) $(Z \vee (\neg Z \vee W)) \wedge \neg(W \wedge U)$
   (f) $(L \to (M \to N)) \to (L \to (M \to N))$

8. Using the expressions determined from Question Four, calculate the valuations of the following propositional formulae and determine whether they are tautologies or contradictions.

   (a) $v(P \vee \neg P)$
   (b) $v(P \wedge \neg P)$
   (c) $v(P \to (Q \to P))$
   (d) $v((P \to Q) \vee (Q \to P))$

9. In class, we introduced propositional logic with each of the logical connectives $\neg, \vee, \wedge, \to$, and $\leftrightarrow$.

   We commented that $\leftrightarrow$ could be *defined* in terms of $\wedge$ and $\to$. This question will explore further simplifications to the language of propositional logic.

   In fact, it is sufficient to introduce the truth tables of $\neg$ and $\vee$ alone. Determine well-formed formulae in $\neg, \vee$ that are logically equivalent to the following formulae:

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

10. **Universal Connectives** Actually, one connective is sufficient. Let us define the logical connective NAND, denoted $\otimes$, with the following truth table:

A	B	$A \otimes B$
1	1	0
1	0	1
0	1	1
0	0	1

    We can think of this as not-and.

    (a) Write the $\neg$ connective in terms of NANDs alone.
    (b) Write the $\vee$ connective in terms of NANDs alone.

    NOR, not-or, is another universal connective.

OPTIONAL EXTRAS

Logic and computation are connected in a number of ways. In class, we are exploring how logicians and mathematicians provided a lot of the original work towards understanding computation. In this tutorial you will see how propositional logic plays an important role in the design of modern CPUs. This is outside the course syllabus. If you would like to read more about this, then you may read Foundations of Computation Section 1.3.

Binary functions are functions which have binary number inputs and outputs. We can use truth-tables to specify the output of a Boolean function for the different values of its input bits. For example:

ZERO(b_1) is defined by the table

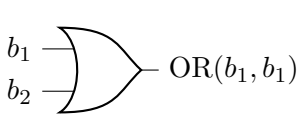
b_1	ZERO
0	0
1	0

AND(b_1, b_2) is defined by the table

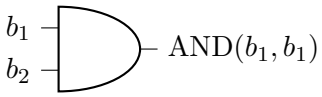
b_1	b_2	AND
0	0	0
1	0	0
0	1	0
1	1	1

One can think of these functions as *gates* with input/output pins. Circuits can be created by connecting output pins of one gate, to input pins of other gates; that is, by composing binary functions. These circuits represent new binary functions.

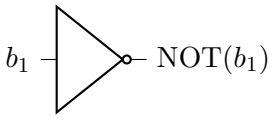
The propositional connectives can be thought of as binary functions and we represent them using the following *logic gates*.



OR-gate



AND-gate



NOT-gate

Each logic gate has input pins and output pins. When combined in the right way, they can be used to design circuits to mimic binary functions. In fact, the *arithmetic logic unit* in modern cpu design uses networks of such gates to achieve all the arithmetic and logical calculations it needs.

1. Using the three logic gates defined above, design circuits that mimic the following binary functions.

   (a) (Half Adder) Give a circuit diagram to calculate the following binary function.

b_1	b_2	sum	carry
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

   This mimics the addition of two bits and keeping track of the carry.

   (b) (Full Adder) Give a circuit diagram to calculate the following binary function.

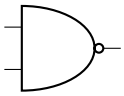
b_1	b_2	c	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

This mimics adding two-bits together with a carry bit.

(c) 2-bit Adder. Design a circuit that takes two 2-bit inputs and outputs their 2-bit sum. Ignore any carry at the end.

(d) 4-bit Adder. Design a circuit that takes two 4-bit inputs and outputs their 4-bit sum. Ignore any carry at the end.

2. The NAND-gate is universal.



NAND-gate

b_1	b_2	NAND
0	0	1
1	0	1
0	1	1
1	1	0

Write each of the following gates with a circuit containing only NAND-gates.

(a) NOT-gate

(b) OR-gate

(c) AND-gate