Lecture 4: Propositional logic

CAB203 Discrete Structures

Matthew McKague

Queensland University of Technology

cab203@qut.edu.au

Outline

Readings

This week

- Pace: 2.1 to 2.5

Next week

- Pace: 3.1 to 3.3
- Lawson: Chapter 2 relates to lectures 4, 5 and 6

Propositional logic

Propositional logic studies:

- Propositions (statements which are true or false)
- Logical connectives that build larger propositions from smaller ones

Logic allows us to determine if a large proposition is true or not based on how it is constructed and the truth value of the smaller pieces.

Outline

Propositions

Logical operators

Formulas

Logical equivalence

Logic and computers

Propositions

A *proposition* is a statement that is either true or false:

- I like tomatoes could be true or false
- All humans are mortal is true
- This sentence is false is neither true or false, so it is not a proposition.

We will often uses symbols p, q etc. to stand in for propositions:

- $p =$ I like tomatoes
- $q =$ all humans are mortal

More examples of propositions

- Propositions from math:
  - $5 \in \{2x : x \in \mathbb{N}\}$
  - $3 \equiv 7 \pmod 4$
- Propositions from the world:
  - Rain comes from peaches
  - Socrates is human
- Complex propositions:
  - It is sunny if and only if it is raining
  - Goats eat grass and goats eat hats

Examples of non-propositions

If we can't assign a truth value, then it isn't a proposition.

- This sentence is false.
- Which way to the bus stop?
- Please make your way to the nearest exit.
- I now pronounce you husband and wife.

Atomic and compound propositions

We distinguish between *atomic* and *compound* propositions. Compound propositions are composed of two or more atomic propositions. Atomic propositions cannot be broken down.

- It is raining is atomic. It can't be broken down.
- It is raining and it is cloudy is a compound proposition. It contains the propositions It is raining and It is cloudy.
- It is raining or snowing is also compound. It contains the propositions It is raining and It is snowing.
- If I hit my head then it will hurt is compound. It contains the propositions I hit my head and my head will hurt.

Atomic and compound proposition examples

- Goats eat grass and hats. (compound)
- The species *Sequoiadendron giganteum* is more commonly known as the giant sequoia. (atomic)
- If you do not have a ticket then you cannot enter. (compound)
- It is either raining or sunny. (compound)

Atomic propositions

- Propositional logic doesn't care about the content of an atomic proposition, only whether it is true or false. So we usually replace them with letters.
- In compound propositions we just care about how they are built, not the content of their atomic propositions.

Outline

Logical operators

We can build compound propositions using atomic propositions and *logical operators* (also called *logical connectives*). Some common operators:

- *NOT* symbolised by $\neg$
- *AND* symbolised by $\wedge$
- *OR* symbolised by $\vee$
- *XOR* symbolised by $\oplus$
- *IF..THEN* symbolised by $\rightarrow$
- *IF AND ONLY IF* symbolised by $\leftrightarrow$

There are 4 possible unary logical operators (like $\neg$) and 16 possible binary logical connectives.

Logical *NOT*

NOT operates on one proposition, giving the negation of the proposition

- $p =$ Socrates is mortal
- $\neg p =$ Socrates is not mortal (informally)
- $\neg p =$ It is not true that Socrates is mortal (more formally)

NOT always gives the exact *logical* opposite. Example:

- $\neg$ He is tall would be He is not tall, which is different from He is short.

Socrates was a Greek philosopher who lived in the 5th century BCE.

Truth tables

We can represent logical values of compound propositions using a *truth table*. A truth table lists all possible truth values of atomic propositions, and the truth value of some compound propositions built using them.

Truth table for *NOT*:

p	$\neg p$
T	F
F	T

AND

$p \wedge q$ is true only when both p and q are true:

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

AND has an (evil?) twin called *NAND*, which plays a special role in computer science because it is *universal, or functionally complete*

AND examples

- All humans are mortal $\wedge$ Socrates is human is true.
- All humans are mortal $\wedge$ Socrates is a teapot is false.
- All humans are spoons $\wedge$ Socrates is human is false.
- All humans are spoons $\wedge$ Socrates is a teapot is false.
- Tomatoes are red $\wedge$ Socrates is human is true.

OR

$p \vee q$ is true only when *at least* one of p and q is true:

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

OR also has an (evil?) twin called *NOR*, which is functionally complete. The Apollo guidance computer was built entirely out of *NOR* gates.

OR examples

- All humans are mortal $\vee$ Socrates is human is true.
- All humans are mortal $\vee$ Socrates is a teapot is true.
- All humans are spoons $\vee$ Socrates is human is true.
- All humans are spoons $\vee$ Socrates is a teapot is false.
- Tomatoes are red $\vee$ Socrates is a teapot is true.

OR is inclusive

English has two different meanings for "or". Compare:

- You can take the bus or the train.
- You can have milk or sugar in your tea.

The first is *exclusive*: you can't take *both* the bus and the train (at the same time). The second is *inclusive*: you *can* have both milk and sugar. In logic, *OR* is always inclusive, but in English the exclusive meaning is more often implied.

XOR

$p \oplus q$ is true only when *exactly* one of p and q is true:

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

We use *XOR* when we want the exclusive meaning of *or*.

$(\{T, F\}, \oplus)$ is a cyclic group, equivalent to arithmetic modulo 2.

XOR examples

- Tomatoes are red $\oplus$ Socrates is a teapot is true.
- Tomatoes are red $\oplus$ Socrates is human is false.
- Tomatoes are blue $\oplus$ Socrates is human is true.
- Tomatoes are blue $\oplus$ Socrates is a teapot is false.

In some of the above cases, Socrates needs to return error 418.

IF..THEN

$p \rightarrow q$ means that q must be true whenever p is. But we don't care when p is false.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

When p is false, then $p \rightarrow q$ is always true.

We sometimes say *p implies q* to mean if p then q.

IF..THEN examples

- Socrates is human $\rightarrow$ Socrates is mortal. is True
- Socrates is human $\rightarrow$ Socrates is a teapot. is False
- Socrates is a teapot $\rightarrow$ Socrates is mortal. is True
- Socrates is a teapot $\rightarrow$ Socrates is an alligator. is True
- Socrates is human $\rightarrow$ Tomatoes are red. is True.
- Socrates is blue $\rightarrow$ Tomatoes are red. is True.

It is important to understand that the truth of $p \rightarrow q$ does not depend on any underlying relationship between p and q, only their truth values.

IF AND ONLY IF

As a shorthand, instead of $(p \to q) \land (q \to p)$ we can write $p \leftrightarrow q$.

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

We say p if and only if q.

IF AND ONLY IF examples

- Socrates is human $\leftrightarrow$ Socrates is mortal. is True.
- Socrates is human $\leftrightarrow$ Socrates is a teapot. is False.
- Socrates is a teapot $\leftrightarrow$ Socrates is mortal. is False.
- Socrates is a teapot $\leftrightarrow$ Socrates is an alligator. is True.
- Socrates is human $\leftrightarrow$ Tomatoes are red. is True.

More complex propositions

We can combine compound propositions using logical operators as well:

- $(p \wedge q) \rightarrow p$
- (Socrates is mortal $\rightarrow$ Socrates is human) $\vee$ Humans are blue

We evaluate the truth by working from the atomic propositions outward. There is an order of operations, but we'll always use parentheses.

Outline

Formulas

A *Boolean formula* is a string of symbols that tells how to build a compound proposition. Formulas are defined by these rules:

- T (true), F (false) and lower case letters are all formulas
- If A and B are formulas then so are:
  - $\neg A$
  - $(A \wedge B)$
  - $(A \vee B)$
  - $(A \oplus B)$
  - $(A \rightarrow B)$
  - $(A \leftrightarrow B)$
- no other strings are formulas

This is an example of a *recursive definition.* Recursive definitions come up frequently in computer science.

Formula examples

Formulas are *well formed* if they conform to the rules, otherwise they are *not well formed.*

Some well formed formulas:

- $\neg p$
- $(p \vee q) \to (q \oplus p)$
- $(T \vee (p \leftrightarrow F))$
- $T \wedge p$

Note that we will often omit outer parentheses.

Some not well formed formulas (non-formulas):

- $pq\neg$
- $p \to$
- $(p\vee)q$

Truth value of forumla

To find the truth value of a formula:

- Fill in the truth value for all variables
- Evaluate logical connectives from innermost parentheses outwards

Eg. when $p = T$ and $q = F$

$$\begin{aligned}(p \vee q) \rightarrow (q \oplus p) &= (T \vee F) \rightarrow (F \oplus T) \\ &= T \rightarrow T \\ &= T\end{aligned}$$

Classifying formulas

Three basic kinds of formulas depending on how they behave when we replace variables with truth values

- *tautologies* are always true
- *contradictions* are always false
- *contingent formulas* can be true or false depending on the variables
- *satisfiable formulas* are either tautologies or contingent formulas

Satisfiability is the classic NP-complete problem. If you can find a fast algorithm for determining if a formula is satisfiable you can win yourself \$1 000 000 USD.

Tautologies

A tautology is always true. Examples:

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

We can see this using a truth table:

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

Contradictions

Contradictions are always false

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

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

Outline

Logically equivalent formulas

Some formulas are *logically equivalent* meaning that they are true at the same time. For example, $A \to B$ is logically equivalent to $\neg A \vee B$.

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

Here the last two columns are identical.
We write $A \to B \equiv \neg A \vee B$.
Saying $A \equiv B$ is the same as saying that $A \leftrightarrow B$ is a tautology.

Logically equivalent formula examples

- $\neg\neg A \equiv A$
- $A \wedge B \equiv B \wedge A$
- $A \vee B \equiv B \vee A$
- $A \vee \neg A \equiv T$
- $A \wedge \neg A \equiv F$
- $A \wedge T \equiv A$
- $A \vee F \equiv A$
- $A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$
- $A \vee (B \vee C) \equiv (A \vee B) \vee C$

Logically equivalent formula examples

- $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
- $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
- $A \vee B \equiv \neg(\neg A \wedge \neg B)$
- $A \wedge B \equiv \neg(\neg A \vee \neg B)$
- $A \rightarrow B \equiv \neg A \vee B$
- $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

There are many many more equivalences that describe basic properties of the logical operators.

Using logically equivalent formulas

We can do *substitution* whenever we have two logically equivalent formulas: replace an occurrence of a formula with the thing it is equivalent to

- Suppose $A \equiv B$. Then substituting in B for A in $A \vee C$ we get

$$A \vee C \equiv B \vee C$$

Also:

- If $A \equiv B$ and $B \equiv C$ then
- $A \equiv C$.

By using substitutions, any Boolean formula can be rewritten entirely in *NAND*s, or in *NOR*s.

Using logically equivalent formulas

We can string together equivalences:

$$\begin{aligned} A \to B &\equiv \neg A \vee B \\ &\equiv B \vee \neg A \\ &\equiv \neg(\neg B) \vee \neg A \\ &\equiv \neg B \to \neg A \end{aligned}$$

We have shown $A \to B \equiv \neg B \to \neg A$.

Note that we are implictly using a property called *transitivity*: if $A \equiv B$ and $B \equiv C$ then $A \equiv C$. We'll discuss this property later.

Outline

Bits and logic

There is a natural correspondence between bits and truth values:

- $0 \equiv F$
- $1 \equiv T$

We can use boolean formulas to describe how bits are manipulated within the computer.

A computer is a logic machine.

Logic and sets

There is a close relationship between logic and set operations. Assuming a universe U:

- Unions are like OR:

$$S \cup T = \{x \in U : (x \in S) \vee (x \in T)\}$$

- Intersections are like AND:

$$S \cap T = \{x \in U : (x \in S) \wedge (x \in T)\}$$

- Complements are like NOT:

$$\overline{S} = \{x \in U : \neg(x \in S)\}$$

In ZF set theory, these set theoretic concepts are defined in terms of logical formulas.

Uses of logic

- Basic vocabulary for mathematics and computer science
- Designing and understanding conditional statements in programming
- Tool for analysing algorithms
- Mathematical underpinning of bit logic, logical circuits, computer architecture

Logic in Python

```
>>> True
True
>>> False
False
>>> True and False
False
>>> True and True
True
>>> True or False
True
>>> not True
False
>>> 3 == 4 - 1
True
>>> not 3 == 4 - 1
False
>>>
```

More logic in Python

```
>>> True ^ False           # XOR is same as bitwise XOR
True
>>> def ifthen(x,y):       # no builtin if..then.
...    return (not x) or y # Build our own from a logical equiv
>>> ifthen(True, False)
False
```