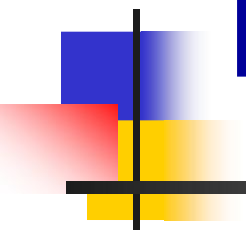

Lecture 1

Course Overview

Chapter 1. The Foundations

1.1 Propositional Logic

What is Mathematics, really?

- It's *not* just about numbers!
- Mathematics is *much* more than that:

> Mathematics is, most generally, the study of any and all *absolutely certain* truths about any and all *perfectly well-defined* concepts.

- These concepts can be *about* numbers, symbols, objects, images, sounds, *anything*!
- It is a way to interpret the world around you.

So, what's *this* class about?

What are "discrete structures" anyway?

- "***Discrete***" - Composed of distinct, separable parts. (Opposite of *continuous*.)

  discrete:*continuous* :: *digital*:*analog*

- "***Structures***" - Objects built up from simpler objects according to some definite pattern.
- "***Discrete Mathematics***" - The study of discrete, mathematical (i.e. well-defined conceptual) objects and structures.

Discrete Objects/Concepts and Structures We Study

- DM PART I
  - Propositions
  - Predicates
  - Proofs
  - Sets
  - Functions
  - Orders of Growth
  - Algorithms
  - Integers
  - Summations
  - Sequences
  - Strings
  - Permutations
  - Combinations
  - Probability
- DM PART II
  - Relations
  - Graphs
  - Trees
  - Boolean Functions / Logic Circuits
  - Automata

Why Study Discrete Math?

- The basis of all of digital information processing is: *Discrete manipulations of discrete structures represented in memory.*
- It's the basic language and conceptual foundation for all of computer science.
- Discrete math concepts are also widely used throughout math, science, engineering, economics, biology, *etc.*, …
- A generally useful tool for rational thought!

Uses for Discrete Math in Computer Science

- Advanced algorithms & data structures
- Programming language compilers & interpreters
- Computer networks
- Operating systems
- Computer architecture
- Database management systems
- Cryptography
- Error correction codes
- Graphics & animation algorithms, game engines, *etc.*…
- *i.e.*, the whole field!

1.1 Propositional Logic

- Logic
  - Study of reasoning.
  - Specifically concerned with whether reasoning is correct.
  - Focuses on the relationship among statements, not on the content of any particular statement.
  - Gives precise meaning to mathematical statements.
- ***Propositional Logic*** is the logic that deals with statements (propositions) and compound statements built from simpler statements using so-called *Boolean connectives.*
- Some applications in computer science:
  - Design of digital electronic circuits.
  - Expressing conditions in programs.
  - Queries to databases & search engines.

Definition of a *Proposition*

Definition: A ***proposition*** (denoted *p*, *q*, *r*, …) is simply:

- a *statement* (*i.e.*, a declarative sentence)
  - *with some definite meaning*,
    (not vague or ambiguous)
- having a *truth value* that's either *true* (**T**) or *false* (**F**)
  - it is **never** both, neither, or somewhere "in between!"
    - However, you might not *know* the actual truth value,
    - and, the truth value might *depend* on the situation or context.
- Later, we will study *probability theory,* in which we assign *degrees of certainty* ("between" **T** and **F**) to propositions.
  - But for now: think True/False only! (or in terms of **1** and **0**)

Examples of Propositions

- It is raining. (In a given situation)
- Beijing is the capital of China. (T)
- 2 + 2 = 5. (F)
- 1 + 2 = 3. (T)
- A fact-based declaration is a proposition, even if no one knows whether it is true
  - 11213 is prime.
  - There exists an odd perfect number.

Examples of Non-Proposition

The following are **NOT** propositions:

- Who's there? (interrogative, question)
- Just do it! (imperative, command)
- La la la la la. (meaningless interjection)
- Yeah, I sorta dunno, whatever... (vague)
- 1 + 2 (expression with a non-true/false value)
- x + 2 = 5 (declaration about semantic tokens of non-constant value)

Truth Tables

- An *operator* or *connective* combines one or more *operand* expressions into a larger expression. (*e.g.*, "+" in numeric expressions.)
- ***Unary*** operators take *one* operand (*e.g.,* −3);
  Binary operators take *two* operands (*e.g.* 3 × 4).
- ***Propositional*** or ***Boolean operators*** operate on propositions (or their truth values) instead of on numbers.
- The ***Boolean domain*** is the set {T, F}. Either of its elements is called a ***Boolean value***.
  An *n*-tuple $(p_1,\ldots,p_n)$ of Boolean values is called a ***Boolean n-tuple***.
- An *n*-operand truth table is a table that assigns a Boolean value to the set of all Boolean *n*-tuples.

Some Popular Boolean Operato

Formal Name	Nickname	Arity	Symbol
Negation operator	NOT	Unary	$\neg$
Conjunction operator	AND	Binary	$\wedge$
Disjunction operator	OR	Binary	$\vee$
Exclusive-OR operator	XOR	Binary	$\oplus$
Implication operator	IMPLIES	Binary	$\rightarrow$
Biconditional operator	IFF	Binary	$\leftrightarrow$

The Negation Operator

- The unary ***negation*** *operator* "¬" (*NOT*) transforms a proposition into its logical *negation*.
- *E.g.* If *p* = "I have brown hair."
  then ¬*p* = "It is not the case that I have brown hair" or "I do **not** have brown hair."
- The *truth table* for NOT:

p	$\neg p$
T	F
F	T

Operand column

Result column

The Conjunction Operator

- The binary ***conjunction*** *operator* "∧" (*AND*) combines two propositions to form their logical *conjunction*.

- *E.g.* If *p* = "I will have salad for lunch." and
  q = "I will have steak for dinner."

  then, *p*∧*q* = "I will have salad for lunch **and** I will have steak for dinner."

Conjunction Truth Table

Operand columns

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

- Note that a conjunction $p_1 \wedge p_2 \wedge \ldots \wedge p_n$ of n propositions will have 2^n rows in its truth table

The Disjunction Operator

- The binary ***disjunction*** *operator* "∨" (*OR*) combines two propositions to form their logical *disjunction*.

- *E.g.* If *p* = "My car has a bad engine." and
  q = "My car has a bad carburetor."

  then, *p*∨*q* = "My car has a bad engine, **or** my car has a bad carburetor."

Meaning is like "and/or" in informal English.

Disjunction Truth Table

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

Note difference from AND

- Note that $p \vee q$ means that p is true, or q is true, **or both** are true!
- So, this operation is also called ***inclusive or***, because it **includes** the possibility that both p and q are true.

The Exclusive-Or Operator

- The binary ***exclusive-or*** *operator* "⊕" (*XOR*) combines two propositions to form their logical "exclusive or"

- *E.g.* If *p* = "I will earn an A in this course." and
  q = "I will drop this course.", then

  p ⊕ *q* = "I will **either** earn an A in this course,
  or I will drop it (**but not both**!)"

Exclusive-Or Truth Table

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

Note difference from OR.

- Note that $p \oplus q$ means that p is true, or q is true, but **not both**!
- This operation is called ***exclusive or***, because it **excludes** the possibility that both p and q are true.

Natural Language is Ambiguou

- Note that the English "or" can be ambiguous regarding the "both" case!
- "Pat is a singer or Pat is a writer." - $\vee$
- "Pat is a man or Pat is a woman." - $\oplus$

p	q	p "or" q
T	T	?
T	F	T
F	T	T
F	F	F

- Need context to disambiguate the meaning!
- For this class, assume "or" means inclusive ($\vee$).

The Implication Operator

- The conditional statement (aka ***implication***) $p \rightarrow q$ states that *p* implies *q.*
- *I.e.*, If *p* is true, then *q* is true; but if *p* is not true, then *q* could be either true or false.
- *E.g.*, let *p* = "You study hard."
  q = "You will get a good grade."
  $p \rightarrow q$ = "If you study hard, then you will get a good grade." (else, it could go either way)
  - *p*: *hypothesis* or *antecedent* or *premise*
  - *q*: *conclusion* or *consequence*

Implication Truth Table

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

The only False case!

- $p \rightarrow q$ is **false** only when p is true but q is **not** true.
- $p \rightarrow q$ does **not** require that p or q **are ever true**!
- *E.g.* "(1=0) → pigs can fly" is TRUE!