Week 2: Logic

Agenda:

- 1.3 Valid and Invalid Arguments
- 1.4 Application: Digital Logic Circuits
- 1.5 Application: Number Systems and Circuits for Addition
- 2.1 Predicates and Quantifiers.

Reading:

- Textbook pages 29–88.

- **Statement**: a declarative sentence that is either true or false, but not both
  - E.g., "The sum of x and y is greater than 0" is **NOT** a statement

- **Argument**: a sequence of statements aimed at demonstrating the truth of an assertion

- **Statement Form**: an expression made up of statement variables and logical connectives

- **Argument Form**: a sequence of statement forms.

  All except the last statement are called Premise / assumption / hypothesis.
  last statement is Conclusion.

- **Valid Argument form**: No matter what statements are substituted for statement variables in the assumption, if the resulting assumptions are true then so is the conclusion.

- E.g.,

$$\begin{array}{ll} & \text{if } p \text{ then } q \\ & p \\ \hline \therefore & q \end{array}$$

- Testing an Argument Form for Validity:
  1. Identify the premises and the conclusion
  2. Construct a truth table showing the truth values of all premises and the conclusion
  3. If there is a row as follows, then it is invalid:
     - All premises are true
     - The conclusion is false

     Otherwise, it is valid.

Truth table for $(p \rightarrow q) \wedge (p) \Rightarrow (q)$:

p	q	$p \rightarrow q$	p	q
0	0	1	0	0
0	1	1	0	1
1	0	0	1	0
1	1	1	1	1

- Testing an Argument Form for Validity:

  A general form: $[p_1 \wedge p_2 \wedge p_3 \wedge \ldots \wedge p_n] \rightarrow q$

  - $p_1, p_2, p_3, \ldots, p_n$ are premises
  - q is the conclusion
  - After the validity is proved, we write $[p_1 \wedge p_2 \wedge p_3 \wedge \ldots \wedge p_n] \Rightarrow q$ '$\Rightarrow$': read as **logically implies**

- The validity can be shown by a truth table
  - We only need to exam those rows in which all premises are true
  - In the other case, '$\Rightarrow$' holds true **vacuously**

- e.g. "If the sum of digits of 12822 is divisable by 3 and the last digit is even then it is divisable by 6".
  "The sum of digits of 12822 is divisable by 3".
  "The last digit is even."
  "Therefore, 12822 is divisable by 6".

Rules of Inference:

- They are (simple enough) valid argument forms

  Syllogism: two premises and a conclusion

- Why these rules?
  - To simplify the other proofs of validity (cannot afford truth tables with large number of variables)

- Several important rules (Table 1.3.1 in Page 40 summarizes 9 rules)

- *Modus Ponens* (Rule of Detachment, Method of Affirming): $[p \wedge (p \rightarrow q)] \Rightarrow q$
- *Modus Tollens* (Rule/Method of Denying): $[(p \rightarrow q) \wedge \sim q] \Rightarrow \sim p$
- Elimination: $[(p \vee q) \wedge \sim p] \Rightarrow q$
- Transitivity: $[(p \rightarrow q) \wedge (q \rightarrow r)] \Rightarrow (p \rightarrow r)$
- Contradiction Rule: $[\sim p \rightarrow F_0] \Rightarrow p$
- generalization: $p \Rightarrow (p \vee q)$
- specialization: $p \wedge q \Rightarrow p$
- Divistion into cases: $[(p \vee q) \wedge (p \rightarrow q) \wedge (q \rightarrow r)] \Rightarrow r$

• Validity is NOT being true; a valid argument may have a false conclusion

• e.g. "if I am teaching 272 I know everything in the world."
"I am teaching 272".
"Therefore, I know everything in the world.".

• More complex example:

$$\begin{array}{r} \sim p \vee q \rightarrow r \\ s \vee \sim q \\ \sim t \\ p \rightarrow t \\ \sim p \wedge r \rightarrow \sim s \\ \therefore \sim q \end{array}$$

$[\sim t \wedge (p \rightarrow t)] \Rightarrow \sim p$ by Modus tollen
$[\sim p] \Rightarrow \sim p \vee q$ by generalization
$[(\sim p \vee q \rightarrow r) \wedge (\sim p \vee q)] \Rightarrow r$ by Modus ponen
$[\sim p \wedge r] \Rightarrow \sim p \wedge r$ by conjunction
$[(\sim p \wedge r \rightarrow \sim s) \wedge (\sim p \wedge r)] \Rightarrow \sim s$ by Modus ponen
$[\sim s \wedge (s \vee \sim q)] \Rightarrow \sim q$ by elimination

Section 1.4: The circuits

- Operations of switches in **series** and **parallel** circuits
  - P and Q in series is equivalent to $P \wedge Q$
  - P and Q in parallel is equivalent to $P \vee Q$
- **Gates**: the basic circuits accepting **input** turning into **output**
  - NOT-gate (one input signal P, one output signal $\sim P$):
  - AND-gate (two input signals P and Q, one output signal $P \wedge Q$)
  - OR-gate (two input signals P and Q, one output signal $P \vee Q$)
- **Combinatorial circuits:**
  - Don't combine two different inputs
  - No feedback (output going back to the input of a gate)
- Computation on circuits:
  - Given a circuit and its input, determine its output
  - Given a circuit, construct its input/output table
  - Given a circuit, find its Boolean expression
  - Given a Boolean expression, construct its circuit
  - Given an input/output table, design its circuit
- Given an input/output table:

Input			Output
P	Q	R	S
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

- Identify the rows with output 1
- For each row with output 1, construct a multi-input AND-gate
- Construct a multi-input OR-gate to have its inputs being the outputs of the constructed AND-gates

  $(P \wedge Q \wedge R) \vee (P \wedge \sim Q \wedge R) \vee (P \wedge \sim Q \wedge \sim R)$

Equivalent circuits:

- Two digital logic circuits are **equivalent** iff their input/output tables are identical
- Two statement forms are **equivalent** iff their truth tables are identical
- This gives another way of **proving** equivalence

  Note: they are essentially the same ...

- This gives a way to simplify the circuit design !!!

  For example, $((P \wedge \sim Q) \vee (P \wedge Q)) \wedge Q$ can be simplified as $P \wedge Q$, an AND-gate

Binary representation of numbers:

- Decimal system: 1,321 = 1 · (1,000) + 3 · (100) + 2 · (10) + 1 · (1)

  or 1,321 = $1 \cdot 10^3 + 3 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0$

Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- Binary representation to use digits 0 and 1 (for those a_i's):

  $1{,}321 = 1 \cdot 2^k + a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \ldots a_0 \cdot 2^0$, for some k

- How do we determine a_i's ?

  a_0 is the remainder of 1,321 divided by 2 (quotient is 660)

  a_1 is the remainder of 660 (the last quotient) divided by 2 (quotient 330)

  ...

Addition of binary numbers:

- The same as we do for adding decimal numbers
- For example, $1101_2 + 111_2 = 10100_2$

$$\begin{array}{ccccc} & 1 & 1 & 1 & \\ & 1 & 1 & 0 & 1_2 \\ & & 1 & 1 & 1_2 \\ \hline 1 & 0 & 1 & 0 & 0_2 \end{array}$$

- Design circuits for addition
  - how do we add single digit numbers?
  - how to take care of **carry**?
  - the table for adding two bits p, q:

p	q	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

  - We can write: $sum = (p \vee q) \wedge \sim (p \wedge q)$
    $carry = p \wedge q$.

  - Now it's easy to build the circuit for sum and carry; this is called half-adder
  - **half-adder**: adds three bits; two bits plus a carry from previous step.

    the circuits, the input/output tables
  - expanding full-adder to any fixed number of digits

Other frequently used representations for numbers:

- The octal system — base-8 representation

  $57_10 = 71_8$

- To transform from binary to octal: consider the bits in groups of three and write the digit (in base 8) for each group; to go from octal to binary do the revers.

- e.g. for 11101011, consider the bits as 011 101 011 then we get: 353_8

- The hexadecimal system — base-16 representation (A, B, C, D, E, F denote 10, 11, 12, 13, 14, 15, respectively)

  $216_{10} = 10 \times 16^1 + 8 \times 16^0 = A8_{16}$.

- To transform between Hex and binary: consider the binary bits in groups of 4:

  e.g. for 11101011 we consider the bits as 1110 1011 and we get: EB_16.

Sectioon 2.1: Predicate logic:

- **Statement**: a declarative sentence that is either true or false, but not both
  - 2 is an integer
  - 2.1 is an integer
  - 20 is an integer
  - $2,000.78$ is an integer
- x is an integer — this is not a statement
- **Predicate**: a declarative sentence that contains a finite number of **variables** and becomes a statement when specific values are substituted for the variables

  Also known as **open statements**, **propositional functions**, and so on
- **Domain** of a predicate variable

Quantifiers:

- Notations for number sets
  - $\mathbb{R}$: real numbers
  - $\mathbb{Q}$: rational numbers
  - $\mathbb{Z}$: integers
  - $\mathbb{R}^+$: positive real numbers
  - $\mathbb{R}^{\geq 0}$: nonnegative real numbers
- **Existential** quantifier:

  There is/exists an x, or there is/exists at least one x, or $\exists x,\ p(x)$

E.g., $\exists x \in \mathcal{Z},\ 2x^2 - x - 3 = 0$

- **Universal** quantifier:

  For all x, or for any x, or $\forall x,\ p(x)$

  E.g., $\forall x \in \mathcal{Z},\ 2x^2 - x - 3 > 0$

Quantified Statements:

- **Existential** statement:

  "$\exists x \in \mathcal{Z},\ 2x^2-x-3 = 0$" is true, since $x = -1$ makes $2x^2-x-3 = 0$

- **Universal** statement:

  "$\forall x \in \mathcal{Z},\ 2x^2-x-3 > 0$" is false, since $x = 0$ makes $2x^2-x-3 < 0$

  Such an "$x = 0$" is called a **counterexample** to the universal statement

- Clearly, $\forall x\ p(x) \Rightarrow \exists x\ p(x)$

Relations between quantifiers and negation:

- Predicate $p(x)$, its negation is $\sim p(x)$

- Four quantified statements:

  - $\exists x\ p(x)$ negation becomes $\forall x\ \ \sim p(x)$
    e.g. "There is somebody in this room who is at least 6 feet."
    Negation: "Everybody in this room is shorter than 6".

  - $\forall x\ p(x)$ negation becomes $\exists x\ \ \sim p(x)$ e.g. "All CS courses involve programming".
    Negation: "There is a CS course that does not involve programming.".

- Predicates with multiple variables:

  $\exists x, y,\ P(x,y) \equiv \exists x \exists y,\ P(x,y) \equiv \exists y \exists x,\ P(x,y)$

$\forall x, y,\ P(x, y) \equiv \forall x \forall y,\ P(x, y) \equiv \forall y \forall x,\ P(x, y)$

e.g. $\forall x, y \in \mathbb{R},\ (x + y)^2 = x^2 + 2xy + y^2$.

- BUT $\forall x \exists y,\ P(x, y) \not\equiv \exists y \forall x,\ P(x, y)$

  e.g. $\forall x \exists y,\ x + y = 10$ means for every value of x we can choose y to be $10 - x$, and thus $x + y = 10$ will be true. But it is not true that there is a value of y s.t. for every value of x that we choose $x + y = 10$.

- Note: $\exists x \forall y,\ P(x, y) \Rightarrow \forall y \exists x,\ P(x, y)$;