Chapter 2 Review

Sunday, April 18, 2021 12:09 PM

Chapter 2: The Logic of Compound Statements

Section 2.1: Logical Form and Logical Equivalence

Statement

- Def: A statement (or proposition) is a sentence that is true of false but not both

Symbols

Negation of *p*	$\sim p$	"not p"
Conjunction of *p* and *q*	$p \wedge q$	"p and q"/"p but q"
Disjunction of *p* and *q*	$p \vee q$	"p or q"

Note: "neither p nor q" ≡ "not p and not q"

Notations of Inequalities

x <= a	x < a or x = a
a <= x <= b	a <= x and x <= b

Truth Tables

Truth Table for $\sim p$

p	$\sim p$
T	F
F	T

Truth Table for $p \wedge q$

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

Truth Table for $p \vee q$

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

Statement Form

- Def: A **statement form** (or **proposition form**) is an expression made up of statement variables and logical connectives that becomes a statement when values (true or false) are substituted into the variables.
- The truth table for a given statement form displays the truth values that correspond to all possible combination of truth values for its statement variables.

Logical Equivalence (≡)

- Def: Two statement forms are **logically equivalent** IFF they have identical truth values for each possible substitution of statements for their statement variables.

Testing Whether Two Statement Forms P and Q Are Logically Equivalent

1) Construct a truth table with one column for the truth values of P and another column for the truth value of Q

2) Check each combination of truth values of the statement variables to see whether the truth value of P is the same as the truth value of Q
   a) If in each row the truth value of P is the same as the truth value of Q, then P and Q are <u>logically equivalent</u>.
   b) If in some row P has a different truth value from Q, then P and Q are <u>NOT logically equivalent</u>.

Tautology (**t**)
- Def: A **tautology** is a statement form that is <u>always true</u>.

Contradiction (**c**)
- Def: A **contradiction** is a statement form that is <u>always false</u>.

Logical Equivalences (<u>Theorem 2.1.1</u>)

Theorem 2.1.1 Logical Equivalences

Given any statement variables p, q, and r, a tautology **t** and a contradiction **c**, the following logical equivalences hold.

1. *Commutative laws:*	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
2. *Associative laws:*	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
3. *Distributive laws:*	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
4. *Identity laws:*	$p \wedge \mathbf{t} \equiv p$	$p \vee \mathbf{c} \equiv p$
5. *Negation laws:*	$p \vee \sim p \equiv \mathbf{t}$	$p \wedge \sim p \equiv \mathbf{c}$
6. *Double negative law:*	$\sim(\sim p) \equiv p$	
7. *Idempotent laws:*	$p \wedge p \equiv p$	$p \vee p \equiv p$
8. *Universal bound laws:*	$p \vee \mathbf{t} \equiv \mathbf{t}$	$p \wedge \mathbf{c} \equiv \mathbf{c}$
9. *De Morgan's laws:*	$\sim(p \wedge q) \equiv \sim p \vee \sim q$	$\sim(p \vee q) \equiv \sim p \wedge \sim q$
10. *Absorption laws:*	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
11. *Negations of* **t** *and* **c**:	$\sim \mathbf{t} \equiv \mathbf{c}$	$\sim \mathbf{c} \equiv \mathbf{t}$

Section 2.2: Conditional Statements

Conditional (→)
- Def: The **conditional** of q by p is "If p then q" or "p implies q" and is denoted $p \to q$. We call p the **hypothesis** (or **antecedent**) of the conditional and q the **conclusion** (or **consequence**).
- Representation of conditional as disjunction: $p \to q \equiv \sim p \vee q$.

Truth Table for $p \to q$

p	q	$p \to q$
T	T	T
T	F	F
F	T	T
F	F	T

The Negation of a Conditional: $\sim(p \to q) \equiv p \wedge \sim q$.

The Contrapositive of a Conditional
- Def: The contrapositive of a conditional statement of the form "If p then q" / $p \to q$ is "If $\sim q$ then $\sim p$" / $\sim q \to \sim p$.
- A conditional statement is logically equivalent to its contrapositive. Meaning $p \to q \equiv \sim q \to \sim p$.

The Converse and Inverse of a Conditional

Converse of a Conditional

- Def: The converse of the conditional statement $p \to q$ is $q \to p$.

Inverse of a Conditional

- Def: The inverse of the conditional statement $p \to q$ is $\sim p \to \sim q$.

Note: The converse and inverse of a conditional are logically equivalent to each other.
Meaning $q \to p \equiv \sim p \to \sim q$.

Only If

- Def: p **only if** $q \equiv$ "if not q then not p" / $\sim q \to \sim p \equiv$ "if p then q" / $p \to q$.

Biconditional ($\leftrightarrow$)

- Def: The biconditional of p and q is "*p* if, and only if, *q*" and is denoted $p \leftrightarrow q$.

Truth Table for $p \leftrightarrow q$

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

Order of Operations for Logical Operators

Order of Operations for Logical Operators

1. $\sim$ — Evaluate negations first.
2. $\wedge, \vee$ — Evaluate $\wedge$ and $\vee$ second. When both are present, parentheses may be needed.
3. $\to, \leftrightarrow$ — Evaluate $\to$ and $\leftrightarrow$ third. When both are present, parentheses may be needed.

Necessary and Sufficient Conditions

- Def: r is a **sufficient condition** *for s* $\equiv$ "if r then s" / $r \to s$
- Def: r is a **necessary condition** *for s* $\equiv$ "if not r then not s" / $\sim r \to \sim s$
- Def: r is a **necessary and sufficient condition** *for s* $\equiv$ "r if, and only if, s" / $r \leftrightarrow s$

Section 2.3: Valid and Invalid Arguments

Argument Form

- Def: An **argument form** is a sequence of statement forms.
- All statement forms in an argument form, except the final one, are called **premises**.
- The final statement form is called the **conclusion**.

Valid Argument Form

- Def: To say that an argument form is valid means that no matter what particular statements are substituted for the variables in its premises, if the resulting premises are all true then the conclusion is also true.

Testing an Argument Form for Validity

1) Identify the premises and conclusion of the argument form.
2) Construct a truth table showing the truth values of all the premises and the conclusion.
3) Identify the critical row(s), which is a row of the truth table in which all the premises are true.
   a) If there is a critical row in which the conclusion is false, then the argument form is invalid

b) If the conclusion in every critical row is true, then the argument form is valid

Argument

- Def: An **argument** is a sequence of statements.
- All statements in an argument, except the final one, are called **premises**.
- The final statement is called the **conclusion**.

Valid Argument

- Def: To say that an argument is valid means that its form is valid.

Common Valid Argument Forms

Modus Ponens	Modus Tollens
If p then q p $\therefore q$	If p then q $\sim q$ $\therefore \sim p$

Additional Valid Argument Forms / Rules of Inference

Generalization		Specialization		Elimination		Transitivity	Proof by Division Into Cases
p $\therefore p \vee q$	q $\therefore p \vee q$	$p \wedge q$ $\therefore p$	$p \wedge q$ $\therefore q$	$p \vee q$ $\sim q$ $\therefore p$	$p \vee q$ $\sim p$ $\therefore q$	$p \to q$ $q \to r$ $\therefore p \to r$	$p \vee q$ $p \to r$ $q \to r$ $\therefore r$

Converse Error (Fallacy of Affirming the Consequence)

- Def: An underlying fallacy in an invalid argument where the conclusion of the argument would follow from the premises if the premise $p \to q$ were replaced by its converse.

Converse Error Form
$p \to q$ q $\therefore p$

Inverse Error (Fallacy of Denying the Antecedent)

- Def: An underlying fallacy in an invalid argument where the conclusion of the argument would follow from the premises if the premise $p \to$ q were replaced by its inverse.

Inverse Error Form
$p \to q$ $\sim p$ $\therefore \sim q$

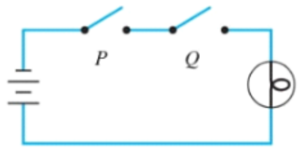
Sound Argument

- Def: An argument is called **sound** IFF it is valid and all its premises are true. An argument that is not sound is called **unsound**.

Contradiction Rule: If you can show that the supposition that statement p is false leads logically to a contradiction $(\sim p \to \mathbf{c})$, then you can conclude that p is true.

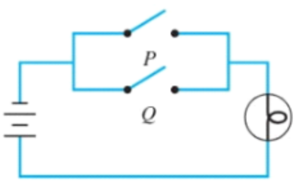
Section 2.4: Digital Logic Circuits

Switches in Series

<table>
<tr><td>Circuit Diagram</td><td>Circuit Behavior Table</td></tr>
<tr><td>
</td><td>
<table>
<tr><th colspan="2">Switches</th><th>Light Bulb</th></tr>
<tr><th>P</th><th>Q</th><th>State</th></tr>
<tr><td>closed</td><td>closed</td><td>on</td></tr>
<tr><td>closed</td><td>open</td><td>off</td></tr>
<tr><td>open</td><td>closed</td><td>off</td></tr>
<tr><td>open</td><td>open</td><td>off</td></tr>
</table>
</td></tr>
</table>

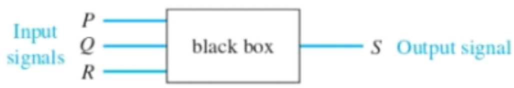
- Corresponds to P ∧ Q

Switches in Parallel

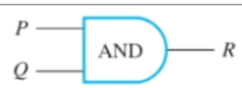
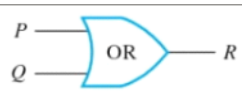
<table>
<tr><td>Circuit Diagram</td><td>Circuit Behavior Table</td></tr>
<tr><td>
</td><td>
<table>
<tr><th colspan="2">Switches</th><th>Light Bulb</th></tr>
<tr><th>P</th><th>Q</th><th>State</th></tr>
<tr><td>closed</td><td>closed</td><td>on</td></tr>
<tr><td>closed</td><td>open</td><td>on</td></tr>
<tr><td>open</td><td>closed</td><td>on</td></tr>
<tr><td>open</td><td>open</td><td>off</td></tr>
</table>
</td></tr>
</table>

- Corresponds to P ∨ Q

Black Box



Gates (Simple Black Box Circuits)

<table>
<tr><td>Type of Gate</td><td>Symbolic Representation</td><td>Action (Input and Output Table)</td><td>Boolean Expression</td></tr>
<tr><td>NOT</td><td>
</td><td>
<table>
<tr><th>Input</th><th>Output</th></tr>
<tr><th>P</th><th>R</th></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td></tr>
</table>
</td><td>R ≡ ~P</td></tr>
<tr><td>AND</td><td>
</td><td>
<table>
<tr><th colspan="2">Input</th><th>Output</th></tr>
<tr><th>P</th><th>Q</th><th>R</th></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
</table>
</td><td>R ≡ P ∧ Q</td></tr>
<tr><td>OR</td><td>
</td><td>
<table>
<tr><th colspan="2">Input</th><th>Output</th></tr>
<tr><th>P</th><th>Q</th><th>R</th></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
</table>
</td><td>R ≡ P ∨ Q</td></tr>
</table>

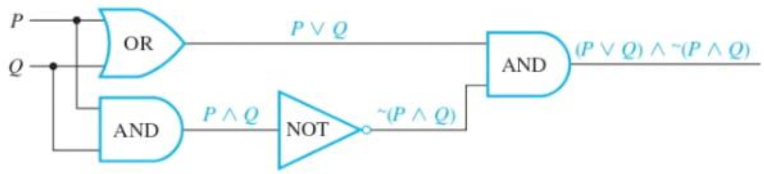
Rules for a Combinational Circuit

- Never combine two input wires
- A single input wire can be split partway and used as input for two separate gates
- An output wire can be used as input
- No output of a gate can eventually feedback into that gate

Boolean Variable and Boolean Expression

- Def: A **Boolean variable** is any variable that can take one of only two values.
- Def: An expression composed of Boolean variables and the connectives ~, ∧, and ∨ is called a **Boolean Expression**.

A Boolean Expression Corresponding to a Circuit Example



Recognizer

- Def: a **recognizer** is a circuit that outputs a 1 for exactly one particular combination of input signals and outputs 0's for all other combinations.

Equivalent Logic Circuits

- Def: Two digital logic circuits are **equivalent** IFF their input/output tables are identical.

NAND and NOR gates

<table>
<tr><td>Type of Gate</td><td>Symbolic Representation</td><td>Action</td><td>Boolean Expression</td></tr>
<tr><td>NAND(| or ↑)</td><td>
</td><td><table>
<tr><td colspan="2">Input</td><td>Output</td></tr>
<tr><td>P</td><td>Q</td><td>R = P | Q</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
</table></td><td>R ≡ ~(P ∧ Q)</td></tr>
<tr><td>NOR (↓)</td><td>
</td><td><table>
<tr><td colspan="2">Input</td><td>Output</td></tr>
<tr><td>P</td><td>Q</td><td>R = P ↓ Q</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
</table></td><td>R ≡ ~(P ∨ Q)</td></tr>
</table>

Section 2.5: Number Systems and Circuits for Addition

Number Systems

Decimal (d_{10})

- Decimal Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Decimal notation is based on the fact that any positive integer can be written uniquely as a sum of products of the form $d \cdot 10^n$, where *n* is a nonnegative integer and each *d* is one of the decimal digits.

Binary (b_2)

- Binary Digits: 0, 1
- Binary notation is based on the fact that any positive integer can be written uniquely as a sum of products of the form $d \cdot 2^n$, where *n* is a nonnegative integer and each *d* is one of the binary digits.

- Note: ln(x)/ln(2) = a means that $2^a = x$

  Powers of 2

Power of 2	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal Form	1024	512	256	128	64	32	16	8	4	2	1

Hexadecimal Notation (h_{16})
- Hexadecimal Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Note that the symbols A, B, C, D, E, and F represent the integers 10 through 15.
- Hexadecimal notation is based on the fact that any positive integer can be written uniquely as a sum of products of the form $d \cdot 16^n$, where *n* is a nonnegative integer and each *d* is one of the hexadecimal digits.

Method to Convert an Integer Between Hexadecimal and Binary

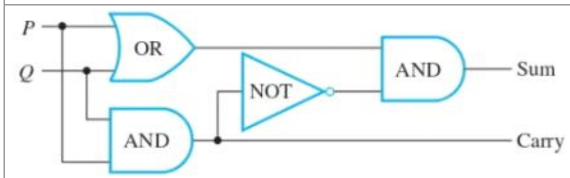
Hexadecimal to Binary
- Write each hexadecimal digit of the integer in 4-bit binary notation.
- Place them together in order.

Binary to Hexadecimal
- Group the digits of the binary number into sets of four, starting from the right and add leading zeros as needed.
- Convert the binary numbers in each set of four into hexadecimal digits.
- Place the hexadecimal digits together in order.

Circuits for Computer Addition

Half-Adder Circuit

Circuit



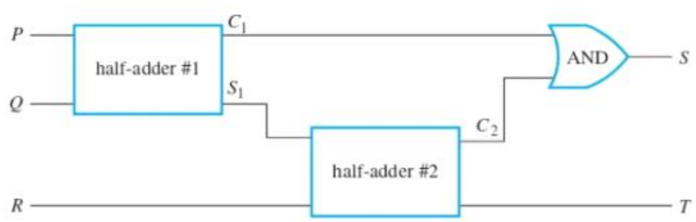
Input/Output Table

P	Q	Carry	Sum
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

- Used to compute the sum of two binary digits *(P* and *Q)*

Full-Adder Circuit

Circuit



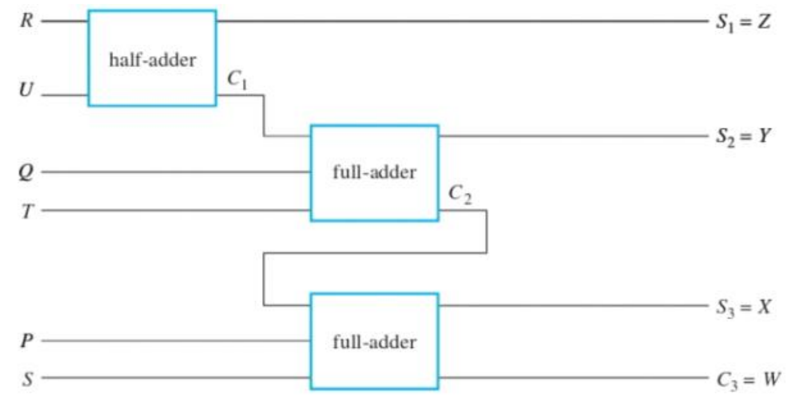
Input/Output Table

P	Q	R	C	S
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

- Use to compute the sum of three binary digits *(P, Q* and *R)*

Parallel Adder Circuit

Circuit



- Used to compute the sum *(WXYZ)* of two three-digit binary numbers *(PQR* and *STU)*

One's Complement

- Def: The one's complement of a given 8-bit binary number a is the result of $(2^8 - 1) - a = 11111111_2 - a$.
- Subtracting an 8-bit binary number a from 11111111_2 just switches all the 0's is a to 1's and all the 1's to 0's.

Two's Complement

- Def: Given a positive integer a, the two's complement of a relative to a fixed bit length n is the n-bit binary representation of $2^n - a$.

Simpler Method of Finding the 8-Bit Two's Complement of a Positive Integer $a \leq 255$

- Write the 8-bit binary representation for a
- Find the one's complement / Flip the bits (that is, switch all the 1's to 0's and all the 0's to 1's)
- Add 1 in binary notation

Method to Find the Decimal Representation of the Integer with a Given 8-Bit Two's Complement

- Find the two's complement of the given two's complement
- Write the decimal equivalent of the result

The 8-Bit Representation of a

$$= \begin{cases} \text{the 8-bit binary representation of } a & \text{if } a \geq 0 \\ \text{the 8-bit binary representation of } 2^8 - |a| & \text{if } a < 0 \end{cases}$$

Computer Addition with Negative Integers

Method to Add Two Integers in the Range -128 to 127 whose Sum is also in the Range -128 to 127

- Convert both integers to their 8-bit representations
- Add the resulting integers using ordinary binary addition
- Truncate any leading 1 (overflow) that occurs in the 2^8th position
- Convert the result back to decimal form (interpreting 8-bit integers with leading 0's as nonnegative and 8-bit integers with leading 1's as negative)

Method to Convert an Integer Between Hexadecimal and Binary

Decimal	Hexadecimal	4-Bit Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hexadecimal to Binary

- Write each hexadecimal digit of the integer in 4-bit binary notation.
- Place them together in order.

Binary to Hexadecimal

- Group the digits of the binary number into sets of four, starting from the right and add leading zeros as needed.
- Convert the binary numbers in each set of four into hexadecimal digits.
- Place the hexadecimal digits together in order.