# Chapter 1. The Foundations

1.2 Propositional Equivalences

1.3 Predicates and Quantifiers

# 1.2 Propositional Equivalence

- A ***tautology*** is a compound proposition that is **true** *no matter what* the truth values of its atomic propositions are!
  - *e.g.* $p \vee \neg p$ ("Today the sun will shine or today the sun will not shine.") [What is its truth table?]
- A ***contradiction*** is a compound proposition that is **false** no matter what!
  - *e.g.* $p \wedge \neg p$ ("Today is Wednesday and today is not Wednesday.") [Truth table?]
- A ***contingency*** is a compound proposition that is neither a tautology nor a contradiction.
  - *e.g.* $(p \vee q) \rightarrow \neg r$

# Logical Equivalence

- Compound proposition $p$ is ***logically equivalent*** to compound proposition $q$, written $p \equiv q$ or $p \Leftrightarrow q$, **iff** the compound proposition $p \leftrightarrow q$ is a tautology.
- Compound propositions $p$ and $q$ are logically equivalent to each other **iff** $p$ and $q$ contain the same truth values as each other in all corresponding rows of their truth tables.

# Proving Equivalence via Truth Tables

- Prove that $\neg(p \wedge q) \equiv \neg p \vee \neg q$. (De Morgan's law)

| $p$ | $q$ | $p \wedge q$ | $\neg p$ | $\neg q$ | $\neg p \vee \neg q$ | $\neg(p \wedge q)$ |
|---|---|---|---|---|---|---|
| T | T | T | F | F | F | F |
| T | F | F | F | T | T | T |
| F | T | F | T | F | T | T |
| F | F | F | T | T | T | T |

- Show that

**Check out the solution in the textbook!**

  - $\neg(p \vee q) \equiv \neg p \wedge \neg q$ (De Morgan's law)
  - $p \to q \equiv \neg p \vee q$
  - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ (distributive law)

# Equivalence Laws

- These are similar to the arithmetic identities you may have learned in algebra, but for propositional equivalences instead.
- They provide a pattern or template that can be used to match part of a much more complicated proposition and to find an equivalence for it and possibly simplify it.

# Equivalence Laws

- *Identity*: $p \wedge \mathbf{T} \equiv p$ $p \vee \mathbf{F} \equiv p$
- *Domination*: $p \vee \mathbf{T} \equiv \mathbf{T}$ $p \wedge \mathbf{F} \equiv \mathbf{F}$
- *Idempotent*: $p \vee p \equiv p$ $p \wedge p \equiv p$
- *Double negation:* $\neg\neg p \equiv p$
- *Commutative:* $p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$
- *Associative:* $(p \vee q) \vee r \equiv p \vee (q \vee r)$
  $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$

# More Equivalence Laws

- *Distributive*: $p \lor (q \land r) \equiv (p \lor q) \land (p \lor r)$
  $p \land (q \lor r) \equiv (p \land q) \lor (p \land r)$
- *De Morgan's*:
  $\neg(p \land q) \equiv \neg p \lor \neg q$
  $\neg(p \lor q) \equiv \neg p \land \neg q$
- *Absorption*
  $p \lor (p \land q) \equiv p$ $\quad$ $p \land (p \lor q) \equiv p$
- *Trivial tautology/contradiction*:
  $p \lor \neg p \equiv \mathbf{T}$ $\quad$ $p \land \neg p \equiv \mathbf{F}$

See Table 6, 7, and 8 of Section 1.2

# Defining Operators via Equivalences

Using equivalences, we can *define* operators in terms of other operators.

- Exclusive or: $p \oplus q \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$
  $p \oplus q \equiv (p \vee q) \wedge \neg(p \wedge q)$
- Implies: $p \rightarrow q \equiv \neg p \vee q$
- Biconditional: $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
  $p \leftrightarrow q \equiv \neg(p \oplus q)$

This way we can "normalize" propositions

# An Example Problem

- Show that $\neg(p \rightarrow q)$ and $p \wedge \neg q$ are logically equivalent.

$\neg(p \rightarrow q)$ [Expand definition of $\rightarrow$]

$\equiv \neg(\neg p \vee q)$ [DeMorgan's Law]

$\equiv \neg(\neg p) \wedge \neg q$ [Double Negation]

$\equiv p \wedge \neg q$

# Another Example Problem

- Check using a symbolic derivation whether

$$\boxed{(p \wedge \neg q) \to (p \oplus r) \equiv \neg p \vee q \vee \neg r}$$

$(p \wedge \neg q) \to (p \oplus r)$ [Expand definition of $\to$]

$\equiv \neg(p \wedge \neg q) \vee (p \oplus r)$ [Expand definition of $\oplus$]

$\equiv \neg(p \wedge \neg q) \vee ( (p \vee r) \wedge \neg(p \wedge r) )$

[DeMorgan's Law]

$\equiv (\neg p \vee q) \vee ( (p \vee r) \wedge \neg(p \wedge r) )$

*cont.*

# Example Continued...

$$\boxed{(p \wedge \neg q) \rightarrow (p \oplus r) \equiv \neg p \vee q \vee \neg r}$$

$(\neg p \vee q) \vee ((p \vee r) \wedge \neg(p \wedge r))$ [$\vee$ Commutative]

$\equiv (q \vee \neg p) \vee ((p \vee r) \wedge \neg(p \wedge r))$ [$\vee$ Associative]

$\equiv q \vee (\neg p \vee ((p \vee r) \wedge \neg(p \wedge r)))$ [Distribute $\vee$ over $\wedge$]

$\equiv q \vee (((\neg p \vee (p \vee r)) \wedge (\neg p \vee \neg(p \wedge r)))$ [$\vee$ Assoc.]

$\equiv q \vee (((\neg p \vee p) \vee r) \wedge (\neg p \vee \neg(p \wedge r)))$ [Trivial taut.]

$\equiv q \vee ((\mathbf{T} \vee r) \wedge (\neg p \vee \neg(p \wedge r)))$ [Domination]

$\equiv q \vee (\mathbf{T} \wedge (\neg p \vee \neg(p \wedge r)))$ [Identity]

$\equiv q \vee (\neg p \vee \neg(p \wedge r))$

*cont.*

# End of Long Example

$$\boxed{(p \wedge \neg q) \rightarrow (p \oplus r) \equiv \neg p \vee q \vee \neg r}$$

$q \vee (\neg p \vee \neg(p \wedge r))$ [DeMorgan's Law]

$\equiv q \vee (\neg p \vee (\neg p \vee \neg r))$ [$\vee$ Associative]

$\equiv q \vee ((\neg p \vee \neg p) \vee \neg r)$ [Idempotent]

$\equiv q \vee (\neg p \vee \neg r)$ [Associative]

$\equiv (q \vee \neg p) \vee \neg r$ [$\vee$ Commutative]

$\equiv \neg p \vee q \vee \neg r$ ■

# Review: Propositional Logic (1.1-1.2)

- Atomic propositions: $p, q, r, \ldots$
- Boolean operators: $\neg \; \wedge \; \vee \; \oplus \; \rightarrow \; \leftrightarrow$
- Compound propositions: $(p \wedge \neg q) \vee r$
- Equivalences: $p \wedge \neg q \Leftrightarrow \equiv \neg(p \rightarrow q)$
- Proving equivalences using:
  - Truth tables
  - Symbolic derivations (series of logical equivalences) $p \equiv q \equiv r \equiv \cdots$

# 1.3 Predicate Logic

- Consider the sentence

  "For every $x$, $x > 0$"

  If this were a true statement about the positive integers, it could not be adequately symbolized using only statement letters, parentheses and logical connectives.

  *The sentence contains two new features: a **predicate** and a **quantifier***

# Subjects and Predicates

- In the sentence "The dog is sleeping":
  - The phrase "the dog" denotes the ***subject*** – the *object* or *entity* that the sentence is about.
  - The phrase "is sleeping" denotes the ***predicate*** – a property that the subject of the statement can have.
- In predicate logic, a ***predicate*** is modeled as a ***proposional function $P(\cdot)$*** from subjects to propositions.
  - $P(x)$ = "$x$ is sleeping" (where $x$ is any subject).
  - $P$(The cat) = "*The cat* is sleeping" (proposition!)

# More About Predicates

- Convention: Lowercase variables $x$, $y$, $z$... denote subjects; uppercase variables $P$, $Q$, $R$… denote propositional functions (or predicates).

- Keep in mind that the *result of applying* a predicate $P$ to a value of subject $x$ is the *proposition*. But the predicate $P$, or the statement $P(x)$ **itself** (*e.g.* $P$ = "is sleeping" or $P(x)$ = "$x$ is sleeping" ) is **not** a proposition.
  - *e.g.* if $P(x)$ = "$x$ is a prime number",

    $P(3)$ is the *proposition* "3 is a prime number."

# Propositional Functions

- Predicate logic *generalizes* the grammatical notion of a predicate to also include propositional functions of **any** number of arguments, each of which may take **any** grammatical role that a noun can take.
  - *e.g.*:
    let $P(x,y,z)$ = "$x$ gave $y$ the grade $z$"
    then if
    $x$ = "Mike", $y$ = "Mary", $z$ = "A",
    then
    $P(x,y,z)$ = "**Mike** gave **Mary** the grade **A**."

# Examples

- Let $P(x)$: $x > 3$. Then
  - $P(4)$ is TRUE/FALSE $\boxed{4 > 3}$
  - $P(2)$ is TRUE/FALSE $\boxed{2 > 3}$
- Let $Q(x, y)$: $x$ is the capital of $y$. Then
  - $Q$(Washington D.C., U.S.A.) is TRUE
  - $Q$(Hilo, Hawaii) is FALSE
  - $Q$(Massachusetts, Boston) is FALSE
  - $Q$(Denver, Colorado) is TRUE
  - $Q$(New York, New York) is FALSE
- Read EXAMPLE 6 (pp.33)
  - If **x > 0** then x:= x + 1 (in a computer program)