SC/MATH 1090

# 1- Boolean Formulae

Ref: G. Tourlakis, *Mathematical Logic*, John Wiley & Sons, 2008.

York University

Department of Computer Science and Engineering

# Overview

- Boolean syntax
  - Boolean Alphabet
  - Strings
  - Formula Calculation; well-formed-formula (WFF)
  - Parsing (top-down and bottom-up)
  - Removing redundant brackets
  - Complexity of formulae

# Boolean Alphabet

1. Symbols for Boolean or propositional variables

   p, q, r with or without primes or subscripts
   Examples: $p$, $p'$, $p_{123}$, $q''_{45}$

2. Symbols for Boolean constants

   ⊤ called top, verum, or symbol "true"
   ⊥ called bottom, falsum, or symbol "false"

3. Brackets, ( and )

4. Boolean connectives

   $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\equiv$

# Strings or Expressions

- Definition:

  A **<u>string</u>** (or word, or expression) over a <u>given alphabet</u> is any <u>ordered</u> sequence of the alphabet's symbols, written adjacent to each other without any visible separators (no commas or spaces, etc).

- Examples:
  - $(p \vee \bot)$ is a string given Boolean alphabet.
  - (p~q) is not a string given Boolean alphabet.
  - $(p \rightarrow q)$ and $\rightarrow p)q($ are two **different** strings given the Boolean alphabet. Note only the ordering is different.

# Strings (cont.)

- String variables
  - Denoted by A, B, C, etc with or without primes or subscripts
- Concatenation
  - Example: if A is *abc* and B is *de* (given the English alphabet), then **AB** is *abcde*
- Empty string
  - Denoted by $\varepsilon$
  - $A\varepsilon = \varepsilon A = A$
- Substring
  - "B is a substring of A" means that for some string C and D we have A= CBD
  - If B is a substring of A and $B \neq A$, then B is a **proper substring** of A.

# Formula calculation

## Procedural definition

- Formula calculation is any finite (ordered) sequence of strings that we may write respecting the following requirements:
  1. At any step, we may write a Boolean variable or a Boolean constant
  2. At any step, we may write $(\neg A)$, provided we have already written string A in a previous step.
  3. At any step, we may write any of the strings
     $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \equiv B)$
     provided we have already written strings A and B in a previous step.

# Well-formed-formula (wff)

- A string A over the Boolean alphabet is called a **Boolean Expression** or a **well-formed-formula** iff it is a string written at some step of some formula-calculation.
  - Examples:
    $(p \equiv q)$
    $((p \vee r) \rightarrow (\neg q))$
- WFF: set of all well-formed-formulae (wffs)
- Bottom- up parsing of a wff is showing the procedural formula calculation steps.

# Recursive definition of WFF

- The set of all well-formed-formulae is the smallest set of strings, WFF, that satisfies
  1. All Boolean variables (p, q, r, ...) , and constants ($\top$, $\bot$)
  2. If A and B are any strings in WFF, then so are the strings ($\neg$ A), (A $\wedge$ B), (A $\vee$ B), (A $\rightarrow$ B), (A $\equiv$ B)

- Top-down parsing of a wff is showing the recursive formula calculation steps.

- How do we know recursion terminates?

- The two definitions for WFF are equivalent.

# Immediate Predecessors (i.p.)

1. Boolean variables or constants don't have any immediate predecessors

2. A is an immediate predecessor of $(\neg A)$

3. A and B are immediate predecessors of

   $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \equiv B)$

- We will prove later that the i.p.s are unique for each formula.

# Removing brackets

- Redundant brackets
  - Outermost brackets are redundant
  - Any pair of brackets is redundant if its presence can be understood from the priority of the connectives
- Priorities:
  - The order of priorities (decreasing) is agreed to be $\neg$ , $\wedge$ , $\vee$ , $\rightarrow$, $\equiv$
  - For same connectives, the rightmost has the highest priority
- Least parenthesized notation (LPN): writing wff with all redundant brackets removed
  - Note writing wff in LPN is just a short notation and is not a correctly written formula (by formula calculation)

# Complexity

- The **complexity** of a formula is the number of connectives occurring in the formula
- The complexity of Boolean variables and constants is zero (they are also called **atomic** formulae)
- Example
  - Complexity of $((p \vee r) \rightarrow (\neg q))$ is 3