COT 3100

APPLICATIONS OF DISCRETE STRUCTURES

DR. ALPER UNGOR • SPRING 2016 • UNIVERSITY OF FLORIDA

Last Revision: January 10, 2016

Table of Contents

1 Foundations of Logic: Overview **1**
1.1 Propositional Logic 1
1.1.1 Basic Definitions 1
1.1.2 Equivalence Rules & Derivations 4
1.2 Predicate Logic 4
1.2.1 Predicates 4
1.2.2 Quantified Predicate Expressions 4
1.2.3 Equivalences & Derivations 4

Abstract

These notes are intended as a resource for myself; past, present, or future students of this course, and anyone interested in the material. The goal is to provide an end-to-end resource that covers all material discussed in the course displayed in an organized manner. If you spot any errors or would like to contribute, please contact me directly.

1 Foundations of Logic: Overview

- Propositional Logic
  - Basic Definitions
  - Equivalence Rules & Derivations
- Predicate Logic
  - Predicates
  - Quantified Predicate Expressions
  - Equivalences & Derivations

1.1 Propositional Logic

Definition 1.1 (Propositional Logic)**.** The logic of compound statements built from simpler statements using *Boolean Connectives.*

Applications:

- Design of digital electronic circuits
- Expressing conditions in computer programs
- Queries to databses & search engines

1.1.1 Basic Definitions

Definition 1.2 (Proposition)**.** A ***proposition (p, q, r, ...)*** is simply a *statement i.e. a declarative sentence) with a definite meaning*, having a *truth value* that's either *true* or *false* (never both, neither, or somewhere in between).

In *probability theory*, we assign *degrees of certainty* to propositions. For now we will just use True/False.

Examples of Propositions

- "It is raining."

Not propositions:

- "Who's there?" (interrogative, question)

Boolean Operators

- Negation (NOT)
- Conjunction (AND)
- Disjunction (OR)
- Exclusive-Or (XOR)
- Implication (IF)
- Bi-conditional (IFF)

Operators & Connectives (Propositions)

- An *operator* or *connective* combines one or more *operand* expressions into a larger expression (e.g. "+" in numeric expressions).
- *Unary* operators take 1 operand (e.g. -3).
  - The unary *negation operator* "¬" (NOT) transforms a proposition into its logical *negation*
    - Example: If p = "I have brown hair.", then ¬ p = "I do not have brown hair."

p	¬ p
T	F
F	T

- *Binary* operators take 2 operands (e.g. 3 x 4).
  - The Conjunction Operator
    - The *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."

p	q	p ∧ q
T	T	T
T	F	F
F	T	F
F	F	F

    - Note that a conjunction table will have 2^n rows.
    - ¬ and ∧ operations together are universal. i.e., sufficient to express any truth table.
  - The Disjunction Operator
    - The binary *disjunction oeprator* "∨" (OR) combines two propositions to form their logical *disjunction*.

      e.g. p = "That car has a bad engine."
      q = "That car has a bad carburetor."
      p ∨ q = "Either that car has a bad engine, or that car has a bad carburetor."

p	q	p ∨ q
T	T	T
T	F	T
F	T	T
F	F	F

    * p ∨ q means that p is true, or q is true, or both are true.
    * This is a *inclusive or*, because it includes the possibility tht both p and q are true.
    * "¬" and "∨" together are also universal
  - The *Exclusive Or* Operator
    * The binary *exclusive-or operator* "⊕" (XOR) combines two propositions to form their logical "exlusive or" (exjunction?)
      p = "I will earn an A in this course,"
      q = "I will drop this course,"
      p ⊕ q = "I will either earn an A for this course, or I will drop it (but not both!)"
    * p ⊕ q means p is true, or q is true, but **not both!**
    * This is called *exclusive or* because it **excludes** the possibility that both p and q are true.
    * ¬ and ⊕ together are not universal.

- *Propositional or Boolean* operators operate on propositions or truth values instead of on numbers.

Operators & Connectives (Conditional Statments)

- The *Implication* Operator
  - Implication p → q states that p implies q.
  - p is the hypothesis (antecedent or premise)
  - q is the conclusion (consequence)
  - It is **false** only in the case that p is **true**, but q is **false**
  - e.g. p = "I am elected." and q = "Taxes will be lowered."
    p → q = "If I am elected, then taxes will be lowered"
  - Implication Truth Table

p	q	p → q
T	T	T
T	F	F
F	T	T
F	F	T

- The *biconditional* Operator
  - Biconditional p → q states that p is true "if and only if" q is true
  - p → q ∧ q → p
  - This is the opposite of ⊕

- p $\iff$ q means $\neg(p \oplus q)$
- e.g. p = "Taxes will be lowered." and q = "If I am elected."
  p $\iff$ q = "Taxes will be lowered if and only if I am elected"
- Implication Truth Table

p	q	p → q
T	T	T
T	F	F
F	T	F
F	F	T

Inverse, Converse, & Contrapositive Intial: p → q

- Inverse: $\neg p \rightarrow \neg q$
- Converse: $q \rightarrow p$
- Contrapositive $\neg q \rightarrow \neg p$

1.1.2 Equivalence Rules & Derivations

1.2 Predicate Logic

1.2.1 Predicates

1.2.2 Quantified Predicate Expressions

1.2.3 Equivalences & Derivations