

3.1 The Basics

Notebook: Discrete Mathematics [CM1020]

Created: 2019-10-07 2:31 PM

Updated: 2019-11-05 5:39 PM

Author: SUKHJIT MANN

Tags: Conjunction, Disjunction, Negation, Precedence, Proposition, Propositional Logic, Truth

Cornell Notes	Topic: 3.1 The Basics	Course: BSc Computer Science
		Class: Discrete Mathematics-Lecture
		Date: November 05, 2019
Essential Question:		
What is propositional logic and how can we make propositions and/or complex propositions?		
Questions/Cues:		
<ul style="list-style-type: none">• What is propositional logic?• What is a proposition?• What a propositional variable?• What is a truth table?• What is a truth set?• What are compound propositions?• What is negation in terms of proposition?• What is conjunction in terms of proposition?• What is disjunction in terms of proposition?• What is exclusive-or in terms of proposition?• What is the precedence of logical operations in terms of propositions?		
Notes		
<ul style="list-style-type: none">• Propositional logic = branch of logic interested in studying mathematical statements<ul style="list-style-type: none">◦ basis of all reasoning and rules used to construct mathematical theories◦ Original purpose, dating back Aristotle was to model reasoning◦ effectively an algebra of propositions<ul style="list-style-type: none">■ variables are unknown propositions instead of real numbers◦ Operators used are: AND, OR, NOT, IMPLIES, & IF AND ONLY IF, instead of +, -, *, & /◦ Can be used comp circuit design & in prog lang like Prolog• Proposition = declarative sentence, either true or false, but not both<ul style="list-style-type: none">◦ Most basic element of logic, to build our reasoning and logical statements◦ examples. "London is the capital of the United Kingdom", "Madrid is the capital of France"• Propositional Variable = is typically a letter, such as: p, q, r, ...<ul style="list-style-type: none">◦ To avoid writing long & repetitive propositions• Truth table = tabular representation of all possible combinations of its constituent variables		

- for n propositions, create table with 2^n rows and n columns

Here are two propositional variables **p** and **q**:

p	q
FALSE	FALSE
FALSE	TRUE
TRUE	FALSE
TRUE	TRUE

Here are 3 propositional variables **p**, **q** and **r**:

p	q	r
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T

- Truth set = Let p be a proposition on a set S . Truth set of p is set of elements of S for which p is true
 - Usually capital letter to denote a truth set of a proposition, ie. the truth set of a proposition p is denoted as P

Let **S** = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Let **p** and **q** be two propositions concerning an integer n in **S**, defined as follows:

p: n is even

q: n is odd

The truth set of **p** written as **P** is:

P = {2, 4, 6, 8, 10}

The truth set of **q** is:

Q = {1, 3, 5, 7, 9}

- Compound propositions = statements built by combining multiple propositions using certain rules
- Negation (\neg) = Let p be a proposition. Negation of p , denoted by $\neg p$ and read "not p ", is statement: "It is not the case that p ."
 - Truth value of negation of p , $\neg p$, is opposite the truth value of p

Example

- **p**: "John's program is written in Python."
- $\neg p$: "John's program is not written in Python."

p	$\neg p$
F	T
T	F

- Conjunction (\wedge) = Let p and q be propositions. Conjunction of p & q, denoted $p \wedge q$, is the proposition "p and q"
 - Conjunction $p \wedge q$ only true when both p & q are true, and false if not the case

Example :

- **p**: "John's program is written in Python."
- **q**: "John's program has less than 20 lines of code."
- **$p \wedge q$** : "John's program is written in Python and has less than 20 lines of code."

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

- Disjunction (\vee) = let p and q be propositions. Disjunction of p & q, denoted $p \vee q$, is proposition "p or q"
 - Disjunction $p \vee q$ only false when both p & q are false; otherwise true

Example

- **p**: "John's program is written in Python."
- **q**: "John's program has less the 20 lines of code."
- **$p \vee q$** : "John's program is written in Python or has less then 20 lines of code."

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

- Exclusive-or (\oplus) = Let p and q be propositions. Exclusive-or of p & q, denoted by $p \oplus q$, is proposition "p or q (but not both)"
 - Exclusive-or $p \oplus q$ true when p is true & q is false and when p is false & q is true

Example :

- **p**: "John's program is written in Python."
- **q**: "John's program has less the 20 lines of code."
- **$p \oplus q$** : "John's program is written in Python or has less then 20 lines of code, but not both."

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

- Precedence of logical operations = to build complex compound propositions, we need to use parentheses, meaning of propositions is different depending on order in which parentheses are used
 - To reduce # of parentheses, use an order of precedence

Example

- $(p \vee q) \wedge (\neg r)$ is different from $p \vee (q \wedge \neg r)$

Example

- $(p \vee q) \wedge (\neg r)$ can be written simply as $p \vee q \wedge \neg r$

Operator	Precedence
\neg	1
\wedge	2
\vee	3

Given a positive integer n, let's consider the propositions p and q:

- **p**: "n is an even number"
- **q**: "n is less than 10"

P1: n is an even number and is less than 10: $(p \wedge q)$

P2: n is either an even number or is less than 10: $(p \vee q)$

P3: n is either an even number or is less than 10 but not both: $(p \oplus q)$

P4: $\neg p \vee (p \wedge q)$

p	q	$(p \wedge q)$	$(p \vee q)$	$p \oplus q$	$\neg p$	$\neg p \vee (p \wedge q)$
F	F	F	F	F	T	T
F	T	F	T	T	T	T
T	F	F	T	T	F	F
T	T	T	T	F	F	T

Summary

In this week, we learned what propositional logic and a proposition is. Alongside this we looked the various operations to perform on propositions, their precedence and truth tables/truth sets.

