CS 278 – Discrete Mathematics for Computer Science

Chapter 1 - Logic

Logic : the study of formal reasoning
: a tool to develop reasonable conclusions based on a given set of data

Proposition : a statement that is either true or false

Truth value : indicates whether a proposition is true or false.

(A proposition's truth value may be known, unknown, or a matter of opinion.)

In the slides that follow and in the textbook, we use single capital letters for the truth values: T for true and F for false.

In Java programming, the boolean values are complete words, all lowercase, true and false.

Compound proposition : one or more individual propositions connected by logical operations.

Logical operations

^	and	"conjunction"
v	or	"disjunction"... also called "inclusive or"
$\oplus$	xor	"exclusive or"
¬	not	"negation"
$\rightarrow$	implies	"conditional operation"

Truth Table

A truth table shows the value of a compound proposition for every combination of truth values included in the compound proposition.

Let's start with the easiest one, negation. The symbol is ¬

Since negation is a unary operation, it only involves one proposition.

Draw a table with two columns and 3 rows. Label the first column p and the second column ¬p.

Negation

p	¬p
T	F
F	T

p has two possible truth values. Write them in the first column.

¬p is the negation of p. When you read it, you can say "not p". Write the truth values for ¬p in the second column.

^ Conjunction (and)

^ (and)

p	q	p ^ q
T	T	T
T	F	F
F	T	F
F	F	F

Notice how the columns for p and q are filled in.

p ^ q is true only when both p and q are true

For all other combinations, it is false

v Disjunction (inclusive or)

v (or)

p	q	p v q
T	T	T
T	F	T
F	T	T
F	F	F

Notice how the columns for p and q are filled in.

p v q is true when either p or q is true

p v q is false only when p and q are both false

⊕ Disjunction (exclusive or)

⊕ (xor)

p	q	p ⊕ q
T	T	F
T	F	T
F	T	T
F	F	F

Notice how the columns for p and q are filled in.

p ⊕ q is true when p is true, or q is true, but not both

p ⊕ q is false when p and q are both true or both false

⊕ Disjunction (exclusive or)

⊕ (xor)

p	q	p ⊕ q
T	T	F
T	F	T
F	T	T
F	F	F

Notice how the columns for p and q are filled in.

p ⊕ q is true when p is true, or q is true, but not both

p ⊕ q is false when p and q are both true or both false

end of section 1.1

Order of operations

Mathematical operations have an order. Operators in a programming language have an order (often called "operator precedence"). So do the logical operators.

Here's the order of operations:

1. ¬ (not)
2. ^ (and)
3. v (or)
4. ⊕ (xor)
5. ⟶ (implies) ⟷ (iff)

¬ has higher precedence than ^; ^ has higher precedence than v;
and v has higher precedence than ⟶ or ⟷.

Order of operations

As with mathematical or programming expressions, parentheses are used to override the order of operations.

It's a good idea to add extra parentheses to make it clear which operation will be performed first.

Examples:

$\neg(p \wedge \neg r)$

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

Filling in the rows of a truth table

As you saw in the truth tables above, there is a pattern to filling in the rows. If there are n variables in a compound proposition, the truth table will have 2^n rows.

Example: Let's write a truth table for (p v r) ^ ¬q.

- Label the first 3 columns with p, q, and r. (use alphabetical order)
- Label the 4th column with the compound proposition.
- Column (r) alternates T and F
- Column (q) alternates TT FF
- Column (p) alternates TTTT FFFF
- Fill in the truth values for the last column.

p	q	r	(p v r) ^ ¬q
T	T	T	F
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	F
F	T	F	F
F	F	T	T
F	F	F	F

Filling in the rows of a truth table

When filling in the truth table for a complicated compound proposition, it can be helpful to add some intermediate columns.

Below is the same truth table as we saw on the previous page, but with two more columns added: a column for (p v r) and a column for ¬q.

p	q	r	p v r	¬q	(p v r) ^ ¬q
T	T	T	T	F	F
T	T	F	T	F	F
T	F	T	T	T	T
T	F	F	T	T	T
F	T	T	T	F	F
F	T	F	F	F	F
F	F	T	T	T	T
F	F	F	F	T	F

end of section 1.2

Conditional Operation

We've examined the truth tables for ¬ (not), v (or), and ^ (and). Now, let's look at ⟶ (called the "conditional operation" or "implies").

p ⟶ q is read as "if p then q" or in some texts "p implies q".

p	q	p ⟶ q
T	T	T
T	F	F
F	T	T
F	F	T

p is called the hypothesis

q is called the conclusion

p ⟶ q is only false when p is true and q is false

Converse, contrapositive and inverse

Converse, contrapositive, and inverse are 3 types of statements related to the conditional p ⟶ q.

Proposition	p ⟶ q	Example: If it is sunny, I will ride my bike to school.
Converse	q ⟶ p	If I ride my bike to school, it is sunny.
Contrapositive	¬q ⟶ ¬p	If I don't ride my bike to school, then it is not sunny.
Inverse	¬p ⟶ ¬q	If it is not sunny, I will not ride my bike to school.

converse "flip"

inverse "negate both"

contrapositive "flip and negate both"

Biconditional operation

The biconditional operation is written with a double-ended arrow. ↔

p ↔ q is read as "p if and only if q" or in some texts "p is necessary and sufficient for q". You may also see "iff" used as an abbreviation for "if and only if".

p	q	p ↔ q
T	T	T
T	F	F
F	T	F
F	F	T

p ↔ q is false when p and q have different truth values

end of section 1.3

Tautology or Contradiction

- Tautology – a compound proposition that is always true
  Example: p ∨ ¬p
- Contradiction – a compound proposition that is always false
  Example: p ∧ ¬p

Logically Equivalent Propositions

2 compound propositions are logically equivalent if their truth tables are the same.

The symbol for logical equivalence is ≡ (similar to equals but 3 lines)

To show that two propositions are NOT logically equivalent, you only have to find one set of truth values that makes the propositions different.

De Morgan's Law - Version 1

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

not(p or q) is logically equivalent to *not p and not q*

Notice how the middle operation changes from V (or) to ∧ (and).

Also, the negation is moved inside the parentheses onto <u>both</u> propositions.

De Morgan's Law - Version 2

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

not(p and q) is logically equivalent to *not p or not q*

Again...the middle operation changes. This time from $\wedge$ (and) to $\vee$ (or). And the negation is moved inside the parentheses onto both propositions.

end of section 1.4

Symbol for "therefore"

$\therefore$ means "therefore"

It's used at the end of a proof. Basically, it means "we're done."

Using De Morgan's Laws

Let's do an example: Use De Morgan's laws to determine whether two expressions are logically equivalent.

Is ¬(p ∨ ¬q) logically equivalent to ¬p ∧ q?

The first expression looks somewhat like the left side in the 1st version of De Morgan's law. ¬(p ∨ q) ≡ (¬p ∧ ¬q)

So let's start with that.

Apply the law:

¬(p ∨ ¬q) ≡ (¬p ∧ ¬ ¬ q)

Keep the parentheses around it if it helps you.

Apply double negation law: $\neg\neg p \equiv p$

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

$\equiv \neg p \wedge q$

That looks the same as the 2nd expression above.

$\therefore$ The two expressions are logically equivalent.

end of section 1.4

Laws of Propositional Logic

In the previous section, we saw De Morgan's laws. There are several others:

Idempotent Laws: ("idempotent" – doesn't change when a proposition operates on itself)

$p \vee p \equiv p$ If you "or" p with itself...it's still p.

$p \wedge p \equiv p$ If you "and" p with itself...it's still p.

Laws of Propositional Logic

Associative Laws:

"associative" refers to how things are **grouped together** with parentheses

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

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

Notice in each of the laws, the operations are all the same, either all "or" or all "and". It doesn't work if there's a mix of "or" and "and".

Laws of Propositional Logic

Commutative Laws:

"commutative" refers to **changing the order** of the propositions

$$p \vee q \equiv q \vee p$$

$$p \wedge q \equiv q \wedge p$$

Distributive laws:
"distributive" refers to spreading (distributing) an operation over
a compound proposition

1. spreading "or" over "and" (watch parentheses carefully)
   $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

2. spreading "and" over "or" (watch parentheses carefully)
   $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$

Laws of Propositional Logic

Identity laws

$p \vee F \equiv p$ p or False is the same as p

$p \wedge T \equiv p$ p and True is the same as p

Laws of Propositional Logic

Domination Laws:

$p \wedge F \equiv F$ anything "and" false is false

$p \vee T \equiv T$ anything "or" true is true

Laws of Propositional Logic

Double negation law

$\neg\neg p \equiv p$

Laws of Propositional Logic

Complement laws:

$p \wedge \neg p \equiv F$

$\neg T \equiv F$

$p \vee \neg p \equiv T$

$\neg F \equiv T$

Laws of Propositional Logic

Absorption laws:

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

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

Laws of Propositional Logic

Conditional identities:

$p \longrightarrow q \equiv \neg p \vee q$

$p \longleftrightarrow q \equiv (p \longrightarrow q) \wedge (q \longrightarrow p)$

end of section 1.5

Logical Reasoning

Argument – a sequence of propositions (called hypotheses) followed by a single proposition (the conclusion).

- Valid argument – conclusion is true when hypotheses are all true
- Invalid argument – conclusion is false when hypotheses are all true

Logical Reasoning

$p_1, p_2, \ldots p_n$ are the hypotheses
c is the conclusion

The argument is valid if $(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \longrightarrow c$ is a tautology (always true).

Is this argument valid?

One way to decide if an argument is valid is to use a truth table.

Example: p1: $p \longrightarrow q$

p2: $p \vee q$

c: q

Write the truth table.

Look at the rows where both hypotheses are true. (rows 1 and 3)

Is the conclusion true in all of those rows? If yes, the argument is valid.

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

An invalid argument

To show that an argument is invalid, you just have to find one case where all hypotheses are true, but the conclusion is false.

Example: p1: $p \vee q$

p2: p

c: $\neg q$

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

Write the truth table.

Look at the rows where both hypotheses are true. (rows 1 and 2)

Is the conclusion true in all of those rows? If not, the argument is invalid.

This argument is invalid because if p is T and $p \vee q$ is T, then the conclusion is F.

Arguments: English to propositional variables

English: It is raining today.

If it is raining today, I will not ride my bike to school.

∴ I will not ride my bike to school.

Choose variables to represent the propositions:

p: It is raining today.

q: I will not ride my bike to school

Write the argument's **form**, using the variables p and q:

p

$p \longrightarrow q$

$\therefore q$

Valid or Invalid

When deciding whether an argument is valid or invalid, it's the **form** you have to look at.

valid: when all hypotheses are true, the conclusion is true
(for every row in the truth table... meaning, every possible combination of truth values!)

invalid: there is at least one row in the truth table where all hypotheses are true, but the conclusion is false

end of section 1.6

Rules of Inference (RoI)

Shortcut to truth tables ☺ We have 8 rules of inferences.

These are arguments that are already shown to be valid.

If your argument has one of these forms, it's valid.

Form	Name of the rule
p p → q ∴ q	Modus ponens (p is true and p implies q. so, q must be true.)
¬q p → q ∴ ¬p	Modus tollens (q isn't true. p implies q which means p isn't true either.)

Rules of Inference (RoI)

Form	Name of the rule	
p ∴ p ∨ q	Addition	p is true ∴ p or q has to be true
p ∧ q ∴ p	Simplification	p and q is true ∴ p has to be true
p q ∴ p ∧ q	Conjunction	p is true q is true ∴ p and q is true

Rules of Inference (RoI)

Form	Name of the rule	
p → q q → r ∴ p → r	Hypothetical syllogism	p implies q is true q implies r is true ∴ p implies r must be true
p ∨ q ¬p ∴ q	Disjunctive syllogism	p or q is true p isn't true ∴ q must be true
p ∨ q ¬p ∨ r ∴ q ∨ r	Resolution	p or q is true not p or r is true ∴ q or r must be true

Logical Proof

Truth tables are too tedious. We can use the laws of propositional logic and the rules of inference to establish the validity of an argument.

When you write the steps in a proof, you have to write a justification for each step.

1) If the proposition in a step is a hypothesis, just write "hypothesis".
2) If the proposition is a result of applying a law or rule, write the name of the law or rule and the previous steps you're applying it to.
3) The last step in the proof must be the conclusion of the argument.

Logical Proof

Example: Is the following argument valid or invalid?

Argument:	Proof: (remember, justify every step)	
$(p \vee r) \longrightarrow q$	1. r	Hypothesis
$q \longrightarrow t$	2. $p \vee r$	Addition, 1
r	3. $(p \vee r) \longrightarrow q$	Hypothesis
$\therefore t$	4. q	M P 2, 3 (apply modus ponens to steps 2 and 3)
	5. $q \longrightarrow t$	Hypothesis
	6. t	M P 4, 5 (apply modus ponens to steps 4 and 5)

We have arrived at the conclusion of the argument.
The argument is valid.

Logical Proof

Another example: Is the following argument valid or invalid?

Argument:	Proof: (remember, justify every step)		
$p \longrightarrow (q \wedge r)$	1. $\neg q$	Hypothesis	
$\neg q$	2. $\neg q \vee \neg r$	Addition, 1 (the trick is to see that you can put literally *anything* on the right side of the "or")	
$\therefore \neg p$	3. $\neg(q \wedge r)$	De Morgan's Law, 2	
	4. $p \longrightarrow (q \wedge r)$	Hypothesis	
	5. $\neg p$	M T 3, 4 (apply modus tollens to steps 3 and 4)	

We have arrived at the conclusion of the argument.
The argument is valid.

Predicates

Predicate = a statement whose truth value is a function of one or more variables.

Some statements are not propositions because they depend on the value of a variable.

Example: x is an odd number. (depends on the value of x)

We can say the truth of the statement is a function of x.

The expression P(x) is read "P of x".

In order to determine the truth value of "x is an odd number", we have to specify a value for x.

Let P(x) = "x is an odd number" and x = 3. P(3) is true.

Predicates

Domain: the set of all possible values for a variable

To determine whether a statement is a proposition or a predicate, ask yourself if the statement depends on a variable.

Domain: the set of all possible values for a variable
Let P(x) = "x is an odd number". For this predicate, the domain of x is the set of all integers.

Quantifiers

Universal quantifier ∀ means "for every" or "for all"

In order to determine the truth value, the domain must be specified.

A universally qualified statement is true if the predicate is true for every value in the domain.

Example:

domain = all positive integers
universally quantified statement = ∀x (x > 0)
The predicate (x > 0) is true for every positive integer.
∴ The statement is true.

Quantifiers

Counterexample = an element in the domain for which the predicate is false

You only have to show one counterexample to show that a universally qualified statement is false.

Quantifiers

Existential quantifier ∃ means "there exists"

To show that an existentially qualified statement is true, you only have to find one example that makes it true.

To show that an existentially qualified statement is false, well... that's not always possible. Sometimes you can show that it's false for an arbitrary value from the domain.

Quantifiers

Example: Given the domain of the set of all positive integers, determine whether $\exists x\ (x^2 < 0)$ is true or false.

Pick an arbitrary x from the domain. $x > 0$

Square both sides. $x^2 > 0$

All positive integers have a square greater than 0.

This contradicts the statement we were given. Therefore, the statement is false.

Review

Proposition = A statement that is either true or false.

Predicate = A statement whose truth value depends on a variable.

Universal quantifier $\forall$

Existential quantifier $\exists$

end of section 1.8

Quantified Statements

We can use logical operations with ∀ and ∃.

∃x (P(x) ∧ ¬O(x)) *There exists an x such that P(x) and not O(x).*

∀x (P(x) → O(x)) *For every x, P(x) implies O(x).*

Quantified statement = a logical statement that includes a universal or existential quantifier.

Free variable vs Bound variable

P(x) x is a free variable

∃ x P(x) x is a bound variable (bound to ∃)

∀ x Q(x) x is a bound variable (bound to ∀)

A statement is a proposition if it has no free variables.

English to Logical Expression

In an English statement, look for these words to help you decide which quantifier to use:

∀ "Everyone" "All" "Every"

∃ "Someone" "At least one" "There is a" "There is an"

Example:

Predicates: S(x): x was sick yesterday
W(x): x went to work yesterday

English statement: Everyone who did not go to work yesterday was sick.

Rephrase: *If x didn't go to work yesterday, then x was sick.*

Logical expression: $\forall x (\neg W(x) \longrightarrow S(x))$

Logical expression to English

Example:

Given these predicates: A(x): x is on the board of directors
E(x): x earns more than $100,000
W(x): x works more than 60 hrs/wk

Translate $\forall x (W(x) \longrightarrow E(x))$ into English.

If the person works > 60 hrs/wk, then they earn more than $100,000.

Solution: Everyone who works more than 60 hours per week earns more than $100,000.

end of section 1.9

De Morgan's Laws for Quantified Statements

You can negate a quantified statement. Here are De Morgan's laws:

$\neg \forall x\, P(x) \equiv \exists x\, \neg P(x)$

$\neg \exists x\, P(x) \equiv \forall x\, \neg P(x)$

De Morgan's Laws for Quantified Statements

Example: Given predicates P(x): x was given the placebo

D(x): x was given the medication

M(x): x had migraines

Use the sentence "There is a patient who took the medication and had migraines".

1. Translate to a logical expression

   $\exists x\, (D(x) \wedge M(x))$

2. Negate the expression

   $\neg \exists x\, (D(x) \wedge M(x)) \equiv \forall x\, \neg(D(x) \wedge M(x)) \equiv \forall x\, (\neg D(x) \vee \neg M(x))$

3. Translate back to English

   "Every patient either did not take the medication or did not have migraines."

end of section 1.10