Week 4 at a glance

We will be learning and practicing to:

- Translate between different representations to illustrate a concept.
  - Translating between symbolic and English versions of statements using precise mathematical language
  - Translating between truth tables (tables of values) and compound propositions
- Use precise notation to encode meaning and present arguments concisely and clearly
  - Listing the truth tables of atomic boolean functions (and, or, xor, not, if, iff)
  - Defining functions, predicates, and binary relations using multiple representations
- Know, select and apply appropriate computing knowledge and problem-solving techniques. Reason about computation and systems. Use mathematical techniques to solve problems. Determine appropriate conceptual tools to apply to new situations. Know when tools do not apply and try different approaches. Critically analyze and evaluate candidate solutions.
  - Evaluating compound propositions
  - Judging logical equivalence of compound propositions using symbolic manipulation with known equivalences, including DeMorgan's Law
  - Writing the converse, contrapositive, and inverse of a given conditional statement
  - Determining what evidence is required to establish that a quantified statement is true or false
  - Evaluating quantified statements about finite and infinite domains

TODO:

Schedule your Test 1 Attempt 1, Test 2 Attempt 1 times at PrairieTest (http://us.prairietest.com)

You must present a physical university-issued or government-issued ID. Copies, photos, or digital IDs are not accepted, and students without verifiable ID will not be permitted to test.

- Pockets must be empty upon entry. Students may bring only their ID and a pen or pencil into the testing center.

Review quizzes based on Week 2 and Week 3 class material (due Monday 01/26/2026)

Homework 2 due on Gradescope https://www.gradescope.com/ (Thursday 01/29/2026)

Start reviewing for Test 1: **Test1 Attempt 1 is next week**. The test covers material in Weeks 1 through Week 5, corresponding to RQ1, RQ2, RQ3, and RQ4. To study for the exam, we recommend reviewing class notes (e.g. annotations linked on the class website, supplementary videos from the class website) and working multiple variants of review quiz questions. You could also find that reviewing homework (and its posted sample solutions)and extra examples (in lecture notes and discussion examples) and getting feedback (office hours and Piazza) is helpful.

Week 4 Monday: Logical Equivalence and Conditionals

Two compound propositions are logically equivalent when...

p	p	¬(¬p)
T	T	¬(F) = T
F	F	¬(T) = F

(Some) logical equivalences

Can replace p and q with any compound proposition

$\neg(\neg p) \equiv p$ **Double negation**

$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$ **Commutativity** Ordering of terms

$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ **Associativity** Grouping of terms

$p \wedge F \equiv F$ $p \vee T \equiv T$ $p \wedge T \equiv p$ $p \vee F \equiv p$ **Domination** aka short circuit evaluation

$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$ **DeMorgan's Laws**

$(p \wedge q) \vee r \not\equiv p \wedge (q \vee r)$

Input p	Input q	$p \wedge q$	Output $\neg(p \wedge q)$	Output $\neg p \vee \neg q$
T	T	T	F	F ∨ F = F
T	F	F	T	F ∨ T = T
F	T	F	T	T ∨ F = T
F	F	F	T	T ∨ T = T

$(p \to q) \to r \stackrel{?}{\equiv} p \to (q \to r)$

Input		Output				
		Conjunction	Exclusive or	Disjunction	Conditional	Biconditional
p	q	$p \wedge q$	$p \oplus q$	$p \vee q$	$p \to q$	$p \leftrightarrow q$
T	T	T	F	T	T	T
T	F	F	T	T	F	F
F	T	F	T	T	T	F
F	F	F	F	F	T	T
		"p and q"	"p xor q"	"p or q"	"if p then q"	"p if and only if q"

exactly one input is T i.e. inputs are different

i.e. inputs have same value.

The only way to make the conditional statement $p \to q$ false is to have p is T and q is F.

The **hypothesis** of $p \to q$ is p

The **antecedent** of $p \to q$ is p

The **conclusion** of $p \to q$ is q

The **consequent** of $p \to q$ is q.

The **converse** of $p \to q$ is $q \to p$.

$p \to q \not\equiv q \to p$

The **inverse** of $p \to q$ is $\neg p \to \neg q$

The **contrapositive** of $p \to q$ is $\neg q \to \neg p$

The conditional $p \to q$ evaluates to T when p being T guarantees that q is T.
The conditional $p \to q$ evaluates to F when we break that promise.

We can use a recursive definition to describe all compound propositions that use propositional variables from a specified collection. Here's the definition for all compound propositions whose propositional variables are in $\{p, q\}$.

Basis Step:	p and q are each a compound proposition
Recursive Step:	If x is a compound proposition then so is $(\neg x)$ and if x and y are both compound propositions then so is each of $(x \wedge y), (x \oplus y), (x \vee y), (x \to y), (x \leftrightarrow y)$

Order of operations (Precedence) for logical operators:

Negation, then conjunction / disjunction, then conditional / biconditionals.

Example: $\neg p \vee \neg q$ means $(\neg p) \vee (\neg q)$.

(Some) logical equivalences

$p \to q \equiv \neg p \vee q$

$p \to q \equiv \neg q \to \neg p$ **Contrapositive**

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

$\neg(p \leftrightarrow q) \equiv p \oplus q$

$p \leftrightarrow q \equiv q \leftrightarrow p$

$p \oplus q \equiv q \oplus p$

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

Extra examples:

$p \leftrightarrow q$ is not logically equivalent to $p \wedge q$ because when $p=F, q=F$ $p \leftrightarrow q$ is T but $p \wedge q$ is F

$p \to q$ is not logically equivalent to $q \to p$ because when $p=T, q=F$ $p \to q$ is F but $q \to p$ is T.

p	q	$p \leftrightarrow q$	$p \wedge q$	$p \to q$	$q \to p$
T	T				
T	F			F	T
F	T			T	F
F	F	T	F		

Common ways to express logical operators in English:

Negation $\neg p$ can be said in English as

- Not p.
- It's not the case that p.
- p is false.

Conjunction $p \wedge q$ can be said in English as

- p and q.
- Both p and q are true.
- p but q.

Exclusive or $p \oplus q$ can be said in English as

- p or q, but not both.
- Exactly one of p and q is true.

Disjunction $p \vee q$ can be said in English as

- p or q, or both.
- p or q (inclusive).
- At least one of p and q is true.

Conditional $p \rightarrow q$ can be said in English as

- if p, then q.
- p is sufficient for q.
- q when p.
- q whenever p.
- p implies q.
- q follows from p.
- q is necessary for p.
- p only if q.

Biconditional

- p if and only if q.
- p iff q.
- If p then q, and conversely.
- p is necessary and sufficient for q.

Week 4 Wednesday: Translation, Predicates, and Quantifiers

Translation: Express each of the following sentences as compound propositions, using the given propositions.

"A sufficient condition for the warranty to be good is that you bought the computer less than a year ago"

w is "the warranty is good"
b is "you bought the computer less than a year ago"

$b \rightarrow w$

"Whenever the message was sent from an unknown system, it is scanned for viruses."

s is "The message is scanned for viruses"
u is "The message was sent from an unknown system"

$u \rightarrow s$

Alternate translation:

k: "The message was sent from a known system"

$\neg k \rightarrow s$

"I will complete my to-do list only if I put a reminder in my calendar"

d is "I will complete my to-do list"
c is "I put a reminder in my calendar"

$d \rightarrow c$

Equivalently: "If I don't put a reminder in my calendar, then I won't complete my to-do list"

$\neg c \rightarrow \neg d$ contrapositive logically equivalent

Different from: "I will complete my to-do list if and only if I put a reminder in my calendar"

$d \leftrightarrow c$, $c \leftrightarrow d$, $c \rightarrow d \wedge d \rightarrow c$

(at least one)

Definition: A collection of compound propositions is called **consistent** if there is an assignment of truth values to the propositional variables that makes each of the compound propositions true.

Consistency:

Whenever the system software is being upgraded, users cannot access the file system. If users can access the file system, then they can save new files. If users cannot save new files, then the system software is not being upgraded.

1. Translate to symbolic compound propositions

Atomic sentences as propositional variable

u = the system software is being upgraded

a = users can access the system

n = users can save new files.

Compound propositions

$u \rightarrow \neg a$

$a \rightarrow n$

$\neg n \rightarrow \neg u$

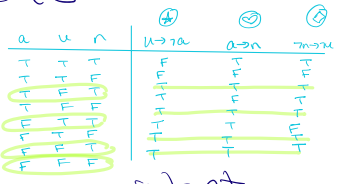
(logically equivalent to $a \rightarrow \neg u$)

2. Look for some truth assignment to the propositional variables for which all the compound propositions output T

Try to make all hyp F: $u = F$ $a = F$ $n = T$

Try to make all conc T: $u = F$ $a = F$ $n = T$

This truth assignment leads to ★ = T, ♡ = T and ◇ = T



extra practice truth table...

So the system ★, ♡, ◇ is consistent.

Another truth assignment that can (alternatively) demonstrate that the collection of compound propositions ★, ♡, ◇ is consistent is $u = T$ $a = F$ $n = T$

Definition: A **predicate** is a function from a given set (domain) to $\{T, F\}$.

A predicate can be applied, or **evaluated** at, an element of the domain.

Usually, a predicate *describes a property* that domain elements may or may not have.

Two predicates over the same domain are **equivalent** means they evaluate to the same truth values for all possible assignments of domain elements to the input. In other words, they are equivalent means that they are equal as functions.

To define a predicate, we must specify its domain and its value at each domain element. The rule assigning truth values to domain elements can be specified using a formula, English description, in a table (if the domain is finite), or recursively (if the domain is recursively defined).

formula for rule

table of values for rule

Input	Output		
	$V(x)$	$N(x)$	$Mystery(x)$
x	$[x]_{2c,3} > 0$	$[x]_{2c,3} < 0$	
000	F	F	T
001	T	F	T
010	T	F	T
011	T	F	F
100	F	T	F
101	F	T	T
110	F	T	F
111	F	T	T

domain elements

example
$V(011)$?
Val in row: T
Formula:
$[011]_{2c,3} > 0 = T$
3

The domain for each of the predicates $V(x)$, $N(x)$, $Mystery(x)$ is $\{000, 001, 010, 011, 100, 101, 110, 111\}$.

Fill in the table of values for the predicate $N(x)$ based on the formula given.

Definition: The **truth set** of a predicate is the collection of all elements in its domain where the predicate evaluates to T.

Notice that specifying the domain and the truth set is sufficient for defining a predicate.

The truth set for the predicate $V(x)$ is $\{001, 010, 011\}$.

The truth set for the predicate $N(x)$ is $\{100, 101, 110, 111\}$.

The truth set for the predicate $Mystery(x)$ is $\{000, 001, 010, 101, 111\}$.

The **universal quantification** of predicate $P(x)$ over domain U is the statement "$P(x)$ for all values of x in the domain U" and is written $\forall x P(x)$ or $\forall x \in U\ P(x)$. When the domain is finite, universal quantification over the domain is equivalent to iterated *conjunction* (ands).

The **existential quantification** of predicate $P(x)$ over domain U is the statement "There exists an element x in the domain U such that $P(x)$" and is written $\exists x P(x)$ for $\exists x \in U\ P(x)$. When the domain is finite, existential quantification over the domain is equivalent to iterated *disjunction* (ors).

An element for which $P(x) = F$ is called a **counterexample** of $\forall x P(x)$.

If there's a counterexample to $\forall x P(x)$ then $\forall x P(x)$ is false.

An element for which $P(x) = T$ is called a **witness** of $\exists x P(x)$.

If there's a witness to $\exists x P(x)$ then $\exists x P(x)$ is true.

Statements involving predicates and quantifiers are **logically equivalent** means they have the same truth value no matter which predicates (domains and functions) are substituted in.

Quantifier version of De Morgan's laws: $\boxed{\neg\forall x P(x) \equiv \exists x\,(\neg P(x))}$ $\boxed{\neg\exists x Q(x) \equiv \forall x\,(\neg Q(x))}$

Examples of quantifications using $V(x), N(x), Mystery(x)$:

True or **False**: $\exists x\ (\ V(x) \wedge N(x)\ )$

$\exists x (V(x) \vee N(x))$

Witness: 001 because 001 is an element in the domain, and Evaluating: $V(001) \vee N(001) = T \vee F = T$.

Conclude: There's at least one element where $V(x) \vee N(x)$ evaluates to true.

True or **False**: $\forall x\ (\ V(x) \rightarrow N(x)\ )$

Counterexample: $x = 001$

$\forall x\ (V(x) \vee N(x))$

Counterexample: 000 because 000 is an element in the domain, and Evaluating: $V(000) \vee N(000) = F \vee F = F$.

Conclude: It's not the case that all elements have $V(x) \vee N(x)$ evaluate to true.

True or **False**: $\exists x\ (\ N(x) \leftrightarrow Mystery(x)\ )$

Witness: $x = 011$

Rewrite $\neg\forall x\ (\ V(x) \oplus Mystery(x)\ )$ into a logical equivalent statement.

Notice that these are examples where the predicates have *finite* domain. How would we evaluate quantifications where the domain may be infinite?

Week 4 Friday: Evaluating Nested Quantifiers

Recall the definitions: The set of RNA strands S is defined (recursively) by:

Basis Step:	$\mathtt{A} \in S, \mathtt{C} \in S, \mathtt{U} \in S, \mathtt{G} \in S$
Recursive Step:	If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

The function *rnalen* that computes the length of RNA strands in S is defined recursively by:

		$rnalen : S$	$\to \mathbb{Z}^+$
Basis Step:	If $b \in B$ then	$rnalen(b)$	$= 1$
Recursive Step:	If $s \in S$ and $b \in B$, then	$rnalen(sb)$	$= 1 + rnalen(s)$

The function *basecount* that computes the number of a given base b appearing in a RNA strand s is defined recursively by:

		$basecount : S \times B$	$\to \mathbb{N}$
Basis Step:	If $b_1 \in B, b_2 \in B$	$basecount(\ (b_1, b_2)\ )$	$= \begin{cases} 1 & \text{when } b_1 = b_2 \\ 0 & \text{when } b_1 \neq b_2 \end{cases}$
Recursive Step:	If $s \in S, b_1 \in B, b_2 \in B$	$basecount(\ (sb_1, b_2)\ )$	$= \begin{cases} 1 + basecount(\ (s, b_2)\ ) & \text{when } b_1 = b_2 \\ basecount(\ (s, b_2)\ ) & \text{when } b_1 \neq b_2 \end{cases}$

Example predicates on S, the set of RNA strands (an infinite set)

$H : S \to \{T, F\}$ where $H(s) = T$ for all s.

Truth set of H is S

$L_{\mathtt{A}} : S \to \{T, F\}$ defined recursively by:

Basis step: $L_{\mathtt{A}}(\mathtt{A}) = T$, $L_{\mathtt{A}}(\mathtt{C}) = L_{\mathtt{A}}(\mathtt{G}) = L_{\mathtt{A}}(\mathtt{U}) = F$

Recursive step: If $s \in S$ and $b \in B$, then $L_{\mathtt{A}}(sb) = L_{\mathtt{A}}(s)$.

Example where $L_{\mathtt{A}}$ evaluates to T is A or AA or ACGU

$L_A(AA) = L_A(A)$ (recstep) $= T$ (basis step)

Example where $L_{\mathtt{A}}$ evaluates to F is G or CAG

Using functions to define predicates:

L with domain $S \times \mathbb{Z}^+$ is defined by, for $s \in S$ and $n \in \mathbb{Z}^+$,

$$L(\ (s,n)\ ) = \begin{cases} T & \text{if } rnalen(s) = n \\ F & \text{otherwise} \end{cases}$$

In other words, $L(\ (s,n)\ )$ means $rnalen(s) = n$

BC with domain $S \times B \times \mathbb{N}$ is defined by, for $s \in S$ and $b \in B$ and $n \in \mathbb{N}$,

$$BC(\ (s,b,n)\ ) = \begin{cases} T & \text{if } basecount(\ (s,b)\ ) = n \\ F & \text{otherwise} \end{cases}$$

In other words, $BC(\ (s,b,n)\ )$ means $basecount(\ (s,b)\ ) = n$

Example where L evaluates to T: $(AC, 2)$ (strand, pos int) Why?

Example where BC evaluates to T: $(AC, C, 1)$ (strand, base, nonneg int) Why?

Example where L evaluates to F: $(GAC, 2)$ Why?

Notice $(\mathbb{Z}, 3)$ is not an example where L evaluates to F.

Example where BC evaluates to F: $(GGA, C, 3)$ Why?

Propositions related to the predicat we just defined:

$\exists t\ BC(t)$ $\exists (s,b,n) \in S \times B \times \mathbb{N}\ (basecount(\ (s,b)\ ) = n)$

In English: There's at least one value (s,b,n) s a strand, b a base, n a nonnegative int where n counts the number of occurrences of b in s.

Witness that proves this existential quantification is true: $(AC, C, 1)$

$\forall t\ BC(t)$ $\forall (s,b,n) \in S \times B \times \mathbb{N}\ (basecount(\ (s,b)\ ) = n)$

In English: For all choices of strand s, base b, nonneg int n this number n equals the number of occurrences of b in s.

Counterexample that proves this universal quantification is false: $(GGA, C, 3)$

New predicates from old

1. Define the **new** predicate with domain $S \times B$ and rule

   $$basecount(\ (s, b)\ ) = 3$$

   The # of occurrences of b in s is exactly 3.

   Example domain element where predicate is T: (ACCC, C)

   ACCC: strand; C: base.

2. Define the **new** predicate with domain $S \times \mathbb{N}$ and rule

   $$basecount(\ (s, \mathtt{A})\ ) = n$$

   The # of occurrences of A in the strand s is n.

   Example domain element where predicate is T: (AAA, 3)

   AAA: strand; 3: nonneg int

3. Define the **new** predicate with domain $S \times B$ and rule

   $$\exists n \in \mathbb{N}\ (basecount(\ (s, b)\ ) = n)$$

   nonneg int

   There is a # that counts the number of occurrences of the base in the strand

   Example domain element where predicate is T: (GGC, A)

   GGC: strand; A: base

4. Define the **new** predicate with domain S and rule

   $$\forall b \in B\ (basecount(\ (s, b)\ ) = 1)$$

   For each base, the # of times this base occurs in the strand is exactly 1.

   Example domain element where predicate is T: ACUG

   strand

Notation: for a predicate P with domain $X_1 \times \cdots \times X_n$ and a n-tuple $(x_1, \ldots, x_n)$ with each $x_i \in X$, we can write $P(x_1, \ldots, x_n)$ to mean $P(\ (x_1, \ldots, x_n)\ )$.

Nested quantifiers

$$\forall s \in S\ \forall b \in B\ \forall n \in \mathbb{N}\ (basecount(\ (s, b)\ ) = n)$$

In English: $\forall s \forall b \forall n\ BC(s, b, n)$

For each strand, each base, and each nonnegative integer, this integer is the number of occurrences of this base in this strand

Counterexample that proves this universal quantification is false: (GGA, C, 3)

$$\forall n \in \mathbb{N}\ \forall s \in S\ \forall b \in B\ (basecount(\ (s, b)\ ) = n)$$

In English: $\forall n \forall s \forall b\ BC(s, b, n)$

For each nonnegative integer, each strand, and each base, this integer is the number of occurrences of this base in this strand

Counterexample that proves this universal quantification is false: (GGA, C, 3)

Alternating nested quantifiers ORDER matters!

$$\forall s \in S \; \exists b \in B \; (\; basecount(\; (s, b) \;) = 3 \;)$$

In English: For each RNA strand there is a base that occurs 3 times in this strand.

Write the negation and use De Morgan's law to find a logically equivalent version where the negation is applied only to the BC predicate (not next to a quantifier).

Is the original statement **True** or **False**?

$$\exists s \in S \; \forall b \in B \; \exists n \in \mathbb{N} \; (\; basecount(\; (s, b) \;) = n \;)$$

In English: There is an RNA strand so that for each base there is some nonnegative integer that counts the number of occurrences of that base in this strand.

Write the negation and use De Morgan's law to find a logically equivalent version where the negation is applied only to the BC predicate (not next to a quantifier).

Is the original statement **True** or **False**?

