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:

Review quiz based on class material each day (due Friday April 26, 2024)

Start reviewing for Test 1, in class next week on Friday May 3, 2024.

Week 4 Monday: Conditionals and Logical Equivalence

Negation: $\neg p$ evaluates to T exactly when p is F

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"

The only way to make the conditional statement $p \to q$ false is to set p to T while 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 $q \to p$

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

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

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.
xor

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

Recall: a contradiction is a compound proposition that always evaluates to F
a tautology is a compound proposition that always evaluates to T

(Some) logical equivalences

Recall: two compound propositions are logically equivalent means their output column of their truth table agree for all inputs.

Notation: $\equiv$ means "is logically equivalent to"

$\not\equiv$ means "is not logically equivalent to"

Can replace p and q with any compound proposition

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

p	$\neg p$	$\neg(\neg p)$
T	F	T
F	T	F

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

$\vee$: "at least one of the inputs is T"
$\wedge$: "both inputs are T"

Also: $p \oplus q \equiv q \oplus p$

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

ORs $\quad$ ANDs

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

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

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

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

$\neg q \to \neg p$ evaluates to F exactly when $\neg q$ is T i.e. q is F and $\neg p$ is F ie. p is T

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

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

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

make conditional F $\quad$ hypothesis is T and conclusion is F.

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

Negation of biconditional is xor

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

Conditional statement is a promise: guarantee that whenever hypothesis is true, then conclusion is also true.

Extra examples:

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

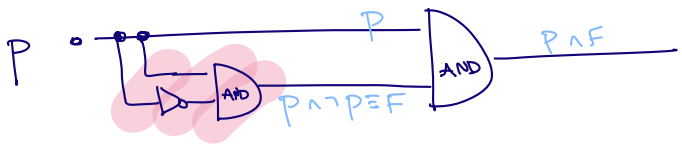
$p \to q$ is not logically equivalent to $q \to p$ because when $p=T, q=F$ $p \to q$ is F but $q \to p$ is T

Extra: distributivity? is xor associative?

$p \wedge F \equiv F$ as a circuit

Circuit implementing $p \wedge F$

Let's use $p \wedge \neg p$ as F



Questions from Chat

- Does the order of the truths and falses impact whether two compound propositions are logically equivalent? Yes! The order in the truth table corresponds to setting the input variables to specific combinations of T and F.
- How might we represent $\equiv$ and $\not\equiv$ in LaTeX?
  \equiv \not\equiv

What's an analogy for biconditional?

a biconditional statement is true exactly when its two inputs have the same truth value as one another

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.
- p is sufficient for q.
- 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.

$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

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

if we bought the computer less than a year ago then the warranty is good.

W

$b \rightarrow w$

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

u

S

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

$u \rightarrow s$

d c

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

$d \rightarrow c$

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

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

$\neg c \rightarrow \neg d$

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:

① Whenever the system software is being upgraded, users cannot access the file system. ② If users can access the file system, then they can save new files. ③ If users cannot save new files, then the system software is not being upgraded.

1. Translate to symbolic compound propositions

u = system software is being upgraded.
a = users can access the file system
n = users can save new files

① Whenever u, $\neg a$ $\quad u \rightarrow \neg a$

② If a, then n $\quad a \rightarrow n$

③ If $\neg n$, then $\neg u$ $\quad \neg n \rightarrow \neg u$

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

Can use guess and check

Can set $u = T$
So have to set $a = F$ so that $u \rightarrow \neg a = T$.
This means can set $n = T$ or $n = F$ so that $a \rightarrow n = T$

Consider $\neg n \rightarrow \neg u = \neg n \rightarrow \neg T = \neg n \rightarrow F$
To ensure conditional statement evaluates to T need hypothesis, $\neg n$, to be F. So $n = T$.

Notice: we've found at least one truth assignment to u, a, n so that ①, ②, ③ all evaluate to T.

Alternatively, draw a truth table with input columns for each propositional variable and an output colum for each compound propositions in collection. Look for (at least) one row in truth table where output columns are T.

Week 4 Wednesday: Predicates and Quantifiers

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

No formula!

Input	Output		
x	$V(x)$ $[x]_{2c,3} > 0$	$N(x)$ $[x]_{2c,3} < 0$	$Mystery(x)$
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 element

$V(010) = T$

$[010]_{2c,3} > 0 = T$

does this relation hold?

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).

Finite domains: T in each row: iterated $\wedge$s

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).

Finite domain: T in at least one row: iterated $\vee$s.

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

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

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: $\boxed{\neg\forall x P(x) \equiv \exists x\,(\neg P(x))}$ $\boxed{\neg\exists x Q(x) \equiv \forall x\,(\neg Q(x))}$

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

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

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

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

predicate

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

Counterexample where $V(x) \to N(x)$ i.e. $V(x) = T$ and $N(x) = F$

consider $x = 001$

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

witness where $N(x)$, $Mystery(x)$ have same truth value.

consider $x = 110$

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

$\exists x \neg(V(x) \oplus Mystery(x))$ or $\exists x\,(\neg V(x) \oplus Mystery(x))$

or $\exists x\,(V(x) \leftrightarrow Mystery(x))$

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

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. (domain, codomain, rule)

Truth set of H is S — not recursive! constant!

Example $AG \in S$, $H(AG) = T$

$F_{\mathtt{A}} : S \to \{T, F\}$ defined recursively by: (domain, codomain, rule)

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

basis case of definition of S was that $A, C, G, U \in S$.

Recursive step: If $s \in S$ and $b \in B$, then $F_{\mathtt{A}}(sb) = F_{\mathtt{A}}(s)$. (strand, base; value of function on shorter strand)

Example where $F_{\mathtt{A}}$ evaluates to T is AG

$F_A(AG) = F_A(A) = T$ (rec step w/ $s=A, b=G$; basis step)

Example where $F_{\mathtt{A}}$ evaluates to F is GA

$F_A(GA) = F_A(G) = F$ (rec step w/ $s=G, b=A$; basis step)

"First (leftmost) base in s is A"

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 $basecount(\ (s,b)\ ) = n$ "the number of occurrences of b in s is n"

Domain element

Example where L evaluates to T: $(AG, 2)$ Why? $rnalen(AG)$ rec step $= 1 + rnalen(A)$ base step $= 1 + 1 = 2$

Example where BC evaluates to T: $(ACA, A, 2)$ Why?

# of occurrences of given base in given strand.

Example where L evaluates to F: $(AG, 1)$ Why?

Example where BC evaluates to F: $(ACA, G, 2)$ Why?

" is equal to "

$t \in S \times B \times \mathbb{N}$

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

In English: There is a choice of strand, s, base, b, nonnegative integer n for which the output of basecount((s,b)) is equal to n.

domain element where predicate evaluates to T.

Witness that proves this existential quantification is true: $(ACA, A, 2)$

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

In English: For each choice of strand s, base b, and nonnegative integer n, the output of basecount((s,b)) is equal to n.

domain element where predicate evaluates to F

Counterexample that proves this universal quantification is false: $(ACA, G, 2)$

Week 4 Friday: Evaluating Nested Quantifiers

New predicates from old

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

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

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

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

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

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

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

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

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

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

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

   Example domain element where predicate is T:

   ACGU UGCA

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:

∀s∀b∀n BC(s,b,n)

For each strand, each base, and each nonneg int, it's that this integer is the number of occurrences of this base in this strand.

Counterexample that proves this universal quantification is false:

(ACGU, A, 2)

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

In English:

∀n∀s∀b BC(s,b,n)

For each nonneg int, each strand, and each base, it's the case that this integer counts the number of occurrences of this base in this strand.

Counterexample that proves this universal quantification is false:

(ACGU, A, 2).

Alternating nested quantifiers

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

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**?

Review Quiz

1. Logical equivalence

   For each of the following propositions, indicate exactly one of:

   - There is no assignment of truth values to its variables that makes it true,
   - There is exactly one assignment of truth values to its variables that makes it true, or
   - There are exactly two assignments of truth values to its variables that make it true, or
   - There are exactly three assignments of truth values to its variables that make it true, or
   - *All* assignments of truth values to its variables make it true.

   (a) $(p \leftrightarrow q) \oplus (p \wedge q)$
   (b) $(p \rightarrow q) \vee (q \rightarrow p)$
   (c) $(p \rightarrow q) \wedge (q \rightarrow p)$
   (d) $\neg(p \rightarrow q)$

2. Translating propositional logic

   (a) Express each of the following sentences as compound propositions, using the given propositions.

   i. "If you try to run Zoom while your computer is running many applications, the video is likely to be choppy and laggy." t is "you run Zoom while your computer is running many applications", c is "the video is likely to be choppy", g is "the video is likely to be laggy"

   $t \rightarrow (c \wedge g)$ $\quad$ $(c \wedge g) \leftrightarrow t$
   $(c \wedge g) \rightarrow t$ $\quad$ $t \oplus (c \wedge g)$

   ii. "To connect wirelessly on campus without logging in you need to use the UCSD-Guest network." c is "connect wirelessly on campus", g is "logging in", and u is "use UCSD-Guest network".

   $c \wedge \neg g \wedge u$ $\quad$ $(c \wedge \neg g) \rightarrow u$
   $(c \wedge \neg g) \vee u$ $\quad$ $u \rightarrow (c \wedge \neg g)$
   $(c \wedge \neg g) \oplus u$ $\quad$ $u \leftrightarrow (c \wedge \neg g)$

(b) For each of the following system specifications, identify the compound propositions that give their translations to logic and then determine if the translated collection of compound propositions is consistent.

i. Specification: If the computer is out of memory, then network connectivity is unreliable. No disk errors can occur when the computer is out of memory. Disk errors only occur when network connectivity is unreliable.
Translation: M = "the computer is out of memory"; N = "network connectivity is unreliable"; D = "disk errors can occur".

$\neg M \rightarrow N$
$\neg D \rightarrow M$
$D \rightarrow N$

$M \rightarrow \neg N$
$\neg D \wedge M$
$N \rightarrow D$

$M \rightarrow N$
$M \rightarrow \neg D$
$\neg N \rightarrow \neg D$

ii. Specification: Whether you think you can, or you think you can't - you're right. [1]
Translation: T = "you think you can"; C = "you can".

$T \rightarrow C$
$\neg T \rightarrow \neg C$

$T \wedge C$
$\neg T \wedge \neg C$

$T \rightarrow \neg T$
$C \rightarrow \neg C$

iii. Specification: A secure password must be private and complicated. If a password is complicated then it will be hard to remember. People write down hard-to-remember passwords. If a password is written down, it's not private. The password is secure.
Translation: S = "the password is secure"; P = "the password is private"; C = "the password is complicated"; H = "the password is hard to remember"; W = "the password is written down".

$\neg(P \wedge C) \rightarrow \neg S$
$C \rightarrow H$
$W \wedge H$
$W \rightarrow \neg P$
S

$(P \wedge C) \rightarrow S$
$C \rightarrow H$
$W \rightarrow H$
$W \rightarrow P$
S

$S \rightarrow (P \wedge C)$
$C \rightarrow H$
$H \rightarrow W$
$W \rightarrow \neg P$
S

[1] Henry Ford

3. Evaluating predicates

(a)

Recall the predicates $V(x)$, $N(x)$, and $Mystery(x)$ on domain $\{000, 001, 010, 011, 100, 101, 110, 111\}$ from class. Which of the following is true? (Select all and only that apply.)

i. $(\forall x\ V(x)) \vee (\forall x\ N(x))$
ii. $(\exists x\ V(x)) \wedge (\exists x\ N(x)) \wedge (\exists x\ Mystery(x))$
iii. $\exists x\ (\ V(x) \wedge N(x) \wedge Mystery(x)\ )$
iv. $\forall x\ (\ V(x) \oplus N(x)\ )$
v. $\forall x\ (\ Mystery(x) \rightarrow V(x)\ )$

(b)

Consider the following predicates, each of which has as its domain the set of all bitstrings whose leftmost bit is 1

$E(x)$ is T exactly when $(x)_2$ is even, and is F otherwise

$L(x)$ is T exactly when $(x)_2 < 3$, and is F otherwise

$M(x)$ is T exactly when $(x)_2 > 256$ and is F otherwise.

i. What is $E(110)$?
ii. Why is $L(00)$ undefined?
   A. Because the domain of L is infinite
   B. Because 00 does not have 1 in the leftmost position
   C. Because 00 has length 2, not length 3
   D. Because $(00)_{2,2} = 0$ which is less than 3
iii. Is there a bitstring of width (where width is the number of bits) 6 at which $M(x)$ evaluates to T?

(c)

For this question, we will use the following predicate.

$F_{\mathtt{A}}$ with domain S is defined recursively by:

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

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

Which of the following is true? (Select all and only that apply.)

i. $F_{\mathtt{A}}(\mathtt{AA})$
ii. $F_{\mathtt{A}}(\mathtt{AC})$
iii. $F_{\mathtt{A}}(\mathtt{AG})$
iv. $F_{\mathtt{A}}(\mathtt{AU})$
v. $F_{\mathtt{A}}(\mathtt{CA})$
vi. $F_{\mathtt{A}}(\mathtt{CC})$
vii. $F_{\mathtt{A}}(\mathtt{CG})$
viii. $F_{\mathtt{A}}(\mathtt{CU})$

4. Evaluating nested predicates

(a)

Recall the predicate L with domain $S \times \mathbb{Z}^+$ from class, $L(\ (s,n)\ )$ means $rnalen(s) = n$. Which of the following is true? (Select all and only that apply.)

i. $\exists s \in S\ \exists n \in \mathbb{Z}^+\ L(\ (s,n)\ )$
ii. $\exists s \in S\ \forall n \in \mathbb{Z}^+\ L(\ (s,n)\ )$
iii. $\forall n \in \mathbb{Z}^+\ \exists s \in S\ L(\ (s,n)\ )$
iv. $\forall s \in S\ \exists n \in \mathbb{Z}^+\ L(\ (s,n)\ )$
v. $\exists n \in \mathbb{Z}^+\ \forall s \in S\ L(\ (s,n)\ )$

(b)

Recall the predicate BC with domain $S \times B \times \mathbb{N}$ from class, $BC(\ (s,b,n)\ )$ means $basecount(\ (s,b)\ ) = n$. Match each sentence to its English translation, or select none of the above.

i. $\forall s \in S\ \exists n \in \mathbb{N}\ \forall b \in B\ basecount(\ (s,b)\ ) = n$
ii. $\forall s \in S\ \forall b \in B\ \exists n \in \mathbb{N}\ basecount(\ (s,b)\ ) = n$
iii. $\forall s \in S\ \forall n \in \mathbb{N}\ \exists b \in B\ basecount(\ (s,b)\ ) = n$
iv. $\forall b \in B\ \forall n \in \mathbb{N}\ \exists s \in S\ basecount(\ (s,b)\ ) = n$
v. $\forall n \in \mathbb{N}\ \forall b \in B\ \exists s \in S\ basecount(\ (s,b)\ ) = n$

i. For each RNA strand and each possible base, the number of that base in that strand is a nonnegative integer.
ii. For each RNA strand and each nonnegative integer, there is a base that occurs this many times in this strand.
iii. Every RNA strand has the same number of each base, and that number is a nonnegative integer.
iv. For every given nonnegative integer, there is a strand where each possible base appears the given number of times.
v. For every given base and nonnegative integer, there is an RNA strand that has this base occurring this many times.

Challenge: Express symbolically

There are (at least) two different RNA strands that have the same number of `A`s.

