This week's highlights

- Evaluate the truth value of a compound proposition given truth values of its constituent variables.
- Prove propositional equivalences using truth tables
- Prove propositional equivalences using other known equivalences, e.g.
  - DeMorgan's laws
  - Double negation laws
  - Distributive laws, etc.
- Compute the CNF and DNF of a given compound proposition.
- Define a predicate over a finite domain using a table of values and as properties
- Determine the truth value of the proposition resulting from evaluating a predicate
- Describe the set of domain elements that make a predicate with one input evaluate to true.
- Evaluate universal and existential statements about finite domains (with no quantifier alternations).
- Counterexample and witness-based arguments for predicates with infinite domains
- Practice combinations of $\wedge$, $\rightarrow$ in conjunction with universal and existential quantifiers
- State and apply DeMorgan's law for quantified statements.
- Translate sentences from English to propositional logic using appropriate propositional variables and boolean operators.

- Decide and justify whether or not a collection of propositions is consistent.
- Use predicates with set of tuples as their domain to relate values to one another
- Evaluate nested quantifiers: both alternating and not.

Lecture videos

Week 4 Day 1 YouTube playlist
Week 4 Day 2 YouTube playlist
Week 4 Day 3 YouTube playlist

Monday January 25

Consider the following algorithm to introduce redundancy in a string of 0s and 1s.

Create redundancy by repeating each bit three times

```
1  procedure redun3(a_{k-1} ··· a_0: a binary string)
2  for i := 0 to k − 1
3    c_{3i} := a_i
4    c_{3i+1} := a_i
5    c_{3i+2} := a_i
6  return c_{3k-1} ··· c_0
```

Decode sequence of bits using majority rule on consecutive three bit sequences

```
1  procedure decode3(c_{3k-1} ··· c_0: a binary string whose length is an integer multiple of 3)
2  for i := 0 to k − 1
3    if exactly two or three of c_{3i}, c_{3i+1}, c_{3i+2} are set to 1
4      a_i := 1
5    else
6      a_i := 0
7  return a_{k-1} ··· a_0
```

Give a recursive definition of the set of outputs of the $redun3$ procedure, Out,

Basis step: _____

Recursive step: _____

Consider the message $m = 0001$ so that the sender calculates $redun3(m) = redun3(0001) = 000000000111$.

Introduce ____ errors into the message so that the signal received by the receiver is _____ but the receiver is still able to decode the original message.

Challenge: what is the biggest number of errors you can introduce?

Building a circuit for line 3 in *decode* procedure: given three input bits, we need to determine whether the majority is a 0 or a 1.

c_{3i}	c_{3i+1}	c_{3i+2}	a_i
1	1	1	
1	1	0	
1	0	1	
1	0	0	
0	1	1	
0	1	0	
0	0	1	
0	0	0	

Circuit

Wednesday January 27

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.

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.

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

The domain for each of the predicates $P(x)$, $N(x)$, $Mystery(x)$ is _____

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

The truth set for the predicate $P(x)$ is _____.

The truth set for the predicate $N(x)$ is _____.

The truth set for the predicate $Mystery(x)$ is _____

Definitions (Rosen 4th ed. p. 57)

The **universal quantification** of $P(x)$ is the statement "$P(x)$ for all values of x in the domain" and is written $\forall x P(x)$. An element for which $P(x) = F$ is called a **counterexample** of $\forall x P(x)$.

The **existential quantification** of $P(x)$ is the statement "There exists an element x in the domain such that $P(x)$" and is written $\exists x P(x)$. An element for which $P(x) = T$ is called a **witness** of $\exists x P(x)$.

Example: _____ is a true existential quantification.

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

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

Example: _____ is a false universal quantification. It is logically equivalent to _____

Recall: Each RNA strand is a string whose symbols are elements of the set $B = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$. The **set of all RNA strands** is called S. The function *rnalen* that computes the length of RNA strands in S is:

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

Example predicates on S

$H(s) = T$	Truth set of H is _____
$L_3(s) = \begin{cases} T & \text{if } rnalen(s) = 3 \\ F & \text{otherwise} \end{cases}$	Strand where L_3 evaluates to T is e.g. Strand where L_3 evaluates to F is e.g.
$F_{\mathtt{A}}$ 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 $F_{\mathtt{A}}(sb) = F_{\mathtt{A}}(s)$	Strand where $F_{\mathtt{A}}$ evaluates to T is e.g. Strand where $F_{\mathtt{A}}$ evaluates to F is e.g.
$P_{\mathtt{AUC}}$ is defined as the predicate whose truth set is the collection of RNA strands where the string AUC is a substring (appears inside s, in order and consecutively)	Strand where $P_{\mathtt{AUC}}$ evaluates to T is e. Strand where $P_{\mathtt{AUC}}$ evaluates to F is e.g

Friday January 29

Definition (Rosen p123): The **Cartesian product** of the sets A and B, $A \times B$, is the set of all ordered pairs (a, b), where $a \in A$ and $b \in B$. That is: $A \times B = \{(a, b) \mid (a \in A) \wedge (b \in B)\}$. The Cartesian product of the sets $A_1, A_2, \ldots, A_n$, denoted by $A_1 \times A_2 \times \cdots \times A_n$, is the set of ordered n-tuples $(a_1, a_2, ..., a_n)$, where a_i belongs to A_i for $i = 1, 2, \ldots, n$. That is, $A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i \text{ for } i = 1, 2, \ldots, n\}$

Recall: Each RNA strand is a string whose symbols are elements of the set $B = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$. The **set of all RNA strands** is called S. The function *rnalen* that computes the length of RNA strands in S is:

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

A function *basecount* that computes the number of a given base b appearing in a RNA strand s is:

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

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

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

Element where L evaluates to T: _____

Element where L evaluates to F: _____

BC with domain _____ is defined S and $b \in B$ and $n \in \mathbb{N}$,

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

Element where BC evaluates to T: _

Element where BC evaluates to F: _

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 write $P(x_1, \ldots, x_n)$ to mean $P(\ (x_1, \ldots, x_n)\ )$.

$\exists t\ BC(t)$ In English: _____

Witness that proves this existential quantification is true: _____

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

Counterexample that proves this universal quantification is false: _____

New predicates from old $BC(s, b, n)$ means $basecount(s, b) = n$.

Predicate	Domain	Example domain element where predicate is T
$basecount(s, b) = 3$		
$basecount(s, \mathtt{A}) = n$		
$\exists n \in \mathbb{N}\ (basecount(s, b) = n)$		
$\forall b \in B\ (basecount(s, b) = 1)$		

Alternating quantifiers

$$\forall s \exists n\ BC(s, \mathtt{A}, n)$$

In English: _____

$$\exists n \forall s\ BC(s, \mathtt{U}, n)$$

In English: _____

~~Evaluate each quantified statement as T or F.~~

$\forall s\ \forall b\ \exists n\ BC(s, b, n)$	$\forall s\ \forall n\ \exists b\ BC(s, b, n)$	$\forall b\ \forall n\ \exists s\ BC(s, b, n)$
$\exists s\ \forall b\ \exists n\ BC(s, b, n)$	$\forall s\ \exists n\ \forall b\ BC(s, b, n)$	$\exists b\ \exists n\ \forall s\ BC(s, b, n)$

Extra example: Write the negation of each of the statements above, 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).

Review quiz questions

1. **Monday** Real-life representations are often prone to corruptions. Biological codes, like RNA, may mutate naturally[1] and during measurement; cosmic radiation and other ambient noise can flip bits in computer storage[2]. One way to recover from corrupted data is to exploit redundancy. Consider the following algorithm to introduce redundancy in a string of 0s and 1s.

   Create redundancy by repeating each bit three times

   ```
   1  procedure redun3(a_{k-1} ··· a_0: a binary string)
   2  for i := 0 to k − 1
   3    c_{3i} := a_i
   4    c_{3i+1} := a_i
   5    c_{3i+2} := a_i
   6  return c_{3k-1} ··· c_0
   ```

   Decode sequence of bits using majority rule on consecutive three bit sequences

   ```
   1  procedure decode3(c_{3k-1} ··· c_0: a binary string whose length is an integer multiple of 3)
   2  for i := 0 to k − 1
   3    if exactly two or three of c_{3i}, c_{3i+1}, c_{3i+2} are set to 1
   4      a_i := 1
   5    else
   6      a_i := 0
   7  return a_{k-1} ··· a_0
   ```

   For each of the following, type in your answers precisely including all notational punctuation.

   (a) Give the output of $redun3(100)$.

   (b) If the output of running $redun3$ is 000000111000111, what was its input?

   (c) Give the output of $decode3(100)$.

   (d) How many distinct possible inputs to $decode3$ give the output 01?

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

[1] Mutations of specific RNA codons have been linked to many disorders and cancers.

[2] This RadioLab podcast episode goes into more detail on bit flips: `https://www.wnycstudios.org/story/bit-flip`

$E(x)$ 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.

(a) What is $E(110)$?

(b) Why is $L(00)$ undefined?

   i. Because the domain of L is infinite

   ii. Because 00 does not have 1 in the leftmost position

   iii. Because 00 has length 2, not length 3

   iv. Because $(00)_{2,3} = 0$ which is less than 3

(c) Is there a bitstring of width (where width is the number of bits) 6 at which $M(x)$ evaluates to T?

3. **Friday** Recall that S is defined as the set of all RNA strands, strings made of the bases in $B = \{\mathtt{A}, \mathtt{U}, \mathtt{G}, \mathtt{C}\}$. Define the functions *mutation*, *insertion*, and *deletion* as described by the pseudocode below:

```
1  procedure mutation(b_1 ··· b_n : a RNA strand, k : a positive integer, b : an element of B)
2  for i := 1 to n
3    if i = k
4      c_i := b
5    else
6      c_i := b_i
7  return c_1 ··· c_n  {The return value is a RNA strand made of the c_i values}
```

```
1  procedure insertion(b_1 ··· b_n : a RNA strand, k : a positive integer, b : an element of B)
2  if k > n
3    for i := 1 to n
4      c_i := b_i
5    c_{n+1} := b
6  else
7    for i := 1 to k − 1
8      c_i := b_i
9    c_k := b
10   for i := k + 1 to n + 1
11     c_i := b_{i−1}
12 return c_1 ··· c_{n+1}  {The return value is a RNA strand made of the c_i values}
```

```
1  procedure deletion(b_1 ··· b_n : a RNA strand, k : a positive integer)
2  if k > n
3    m := n
4    for i := 1 to n
5      c_i := b_i
6  else
7    m := n − 1
8    for i := 1 to k − 1
9      c_i := b_i
10   for i := k to n − 1
11     c_i := b_{i+1}
12 return c_1 ··· c_m  {The return value is a RNA strand made of the c_i values}
```

For this question, we will use the following predicates.

$F_{\mathtt{A}}$ 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 $F_{\mathtt{A}}(sb) = F_{\mathtt{A}}(s)$

$P_{\mathtt{AUC}}$ with domain S is defined as the predicate whose truth set is the collection of RNA strands where the string $\mathtt{AUC}$ is a substring (appears inside s, in order and consecutively)

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

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

Mut with domain $S \times S$ is defined by, for $s_1 \in S$ and $s_2 \in S$,

$$Mut(s_1, s_2) = \exists k \in \mathbb{Z}^+ \exists b \in B(\ mutation(s_1, k, b) = s_2\ )$$

Ins with domain $S \times S$ is defined by, for $s_1 \in S$ and $s_2 \in S$,

$$Ins(s_1, s_2) = \exists k \in \mathbb{Z}^+ \exists b \in B(\ insertion(s_1, k, b) = s_2\ )$$

Del with domain $S \times S$ is defined by, for $s_1 \in S$ and $s_2 \in S$,

$$Del(s_1, s_2) = \exists k \in \mathbb{Z}^+(\ deletion(s_1, k) = s_2\ )$$

(a) 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. $F_{\mathtt{A}}(\mathtt{CU})$

(b) 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. $\exists n \in \mathbb{Z}^+\ \forall s \in S\ (L(s,n))$
vi. $\forall s \in S\ \forall n \in \mathbb{Z}^+\ (L(s,n))$

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

i. $\exists s \in S\ Mut(s,s)$

ii. $\forall s \in S\ Mut(s, s)$
iii. $\exists s \in S\ Ins(s, \mathtt{A})$
iv. $\exists s \in S\ Ins(\mathtt{A}, s)$
v. $\exists s \in S\ Del(s, \mathtt{A})$
vi. $\forall s \in S\ Del(s, \mathtt{A})$