Monday October 18

Real-life representations are often prone to corruption. 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 introduce or 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 nonempty bitstring)
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 nonempty bitstring 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 *redun*3 procedure, *Out*,

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 lines 3-6 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

[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`

Definition: 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 that S is defined as the set of all RNA strands, nonempty strings made of the bases in $B = \{\texttt{A}, \texttt{U}, \texttt{G}, \texttt{C}\}$. We define the functions

$$\textit{mutation} : S \times \mathbb{Z}^+ \times B \to S \qquad \textit{insertion} : S \times \mathbb{Z}^+ \times B \to S$$

$$\textit{deletion} : \{s \in S \mid \textit{rnalen}(s) > 1\} \times \mathbb{Z}^+ \to S \qquad \text{with rules}$$

```
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 with n > 1, 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}
```

Trace the pseudocode to find the output of *mutation*($(\texttt{AUC}, 3, \texttt{G})$)

Fill in the blanks so that *insertion*($(\texttt{AUC}, \_, \_)$) = $\texttt{AUCG}$

Fill in the blanks so that *deletion*($(\_\_, \_)$) = $\texttt{G}$

Review

1.

In this question, we will consider how to build a logic circuit with inputs x_0, y_0, x_1, y_1 and output z such that $z = 1$ exactly when $(x_1x_0)_{2,2} < (y_1y_0)_{2,2}$ and $z = 0$ exactly when $(x_1x_0)_{2,2} \geq (y_1y_0)_{2,2}$.

(a) The first step towards designing this logic circuit is to construct its input-output table. How many rows does this table have (not including the header row labelling the columns)?

(b) What is the output for the row whose input values are $x_0 = 0$, $y_0 = 1$, $x_1 = 1$, $y_1 = 0$?

(c) What is the output for the row whose input values are $x_0 = 0$, $y_0 = 1$, $x_1 = 0$, $y_1 = 1$?

2.

Recall the procedures *redun*3 and *decode*3 from class.

(a) Give the output of *redun*3(100).

(b) If the output of running *redun*3 is 000000111000111, what was its input?

(c) Give the output of *decode*3(100).

(d) How many distinct possible inputs to *decode*3 give the output 01?

3.

Recall the procedures *mutation* and *insertion* and *deletion* from class.

(a) Trace the pseudocode to find the output of *mutation*($(\texttt{AUC}, 2, \texttt{U})$)

(b) Trace the pseudocode to find the output of *insertion*($(\texttt{AUC}, 1, \texttt{G})$)

(c) Trace the pseudocode to find the output of *deletion*($(\texttt{AUC}, 1)$)

Wednesday October 20

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).

Input	Output		
	$V(x)$	$N(x)$	$Mystery(x)$
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 $V(x)$, $N(x)$, $Mystery(x)$ is _____.

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 _____.

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

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

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 $\forall x P(x)$.

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

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

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

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

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

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?

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

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

Truth set of H is _____

$F_{\mathtt{A}} : S \rightarrow \{T, F\}$ 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 $F_{\mathtt{A}}(sb) = F_{\mathtt{A}}(s)$.

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

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

Review

1.

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.)

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

(b) $(\exists x\ V(x)) \wedge (\exists x\ N(x)) \wedge (\exists x\ Mystery(x))$

(c) $\exists x\ (\ V(x) \wedge N(x) \wedge Mystery(x)\ )$

(d) $\forall x\ (\ V(x) \oplus N(x)\ )$

(e) $\forall x\ (\ Mystery(x) \rightarrow V(x)\ )$

2.

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.

(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 $(00)_{2,2} = 0$ 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.

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.)

(a) $F_{\mathtt{A}}(\mathtt{AA})$

(b) $F_{\mathtt{A}}(\mathtt{AC})$

(c) $F_{\mathtt{A}}(\mathtt{AG})$

(d) $F_{\mathtt{A}}(\mathtt{AU})$

(e) $F_{\mathtt{A}}(\mathtt{CA})$

(f) $F_{\mathtt{A}}(\mathtt{CC})$

(g) $F_{\mathtt{A}}(\mathtt{CG})$

(h) $F_{\mathtt{A}}(\mathtt{CU})$

Friday October 22

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

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: _____ Why?

Example where BC evaluates to T: _____ Why?

Example where L evaluates to F: _____ Why?

Example where BC evaluates to F: _____ Why?

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

In English:

Witness that proves this existential quantification is true:

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

In English:

Counterexample that proves this universal quantification is false:

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:

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:

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:

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:

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:

Counterexample that proves this universal quantification is false:

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

In English:

Counterexample that proves this universal quantification is false:

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

1.

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.)

(a) $\exists s \in S\ \exists n \in \mathbb{Z}^+\ L(\ (s,n)\ )$

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

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

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

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

2.

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.

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

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

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

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

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

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.