

Thursday March 31

Notation and prerequisites

Term	Notation	Example(s)	We say in English ...
sequence	x_1, \dots, x_n		A sequence x_1 to x_n
summation	$\sum_{i=1}^n x_i$ or $\sum_{i=1}^n x_i$		The sum of the terms of the sequence x_1 to x_n
all reals	\mathbb{R}		The (set of all) real numbers (numbers on the number line)
all integers	\mathbb{Z}		The (set of all) integers (whole numbers including negatives, zero, and positives)
all positive integers	\mathbb{Z}^+		The (set of all) strictly positive integers
all natural numbers	\mathbb{N} or $\mathbb{Z}^{\geq 0}$		The (set of all) natural numbers. Note: we use the convention that 0 is a natural number.
piecewise rule definition	$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$		Define f of x to be x when x is nonnegative and to be $-x$ when x is negative
function application	$f(7)$ $f(z)$ $f(g(z))$		f of 7 or f applied to 7 or the image of 7 under f f of z or f applied to z or the image of z under f f of g of z or f applied to the result of g applied to z
absolute value	$ -3 $		The absolute value of -3
square root	$\sqrt{9}$		The non-negative square root of 9

Mathematical object and data types

A mathematical object is any abstract object arising in mathematics.

What are some examples of mathematical objects?

Computers encode mathematical objects into data types. What are some examples of data types?

Data Types: sets, n -tuples, and strings

Term	Examples: (add additional examples from class)
set unordered collection of elements <i>repetition doesn't matter</i> <i>Equal sets agree on membership of all elements</i>	$7 \in \{43, 7, 9\}$ $2 \notin \{43, 7, 9\}$
n-tuple (also called an n-dimensional vector) ordered sequence of entries with n “slots” ($n > 0$) <i>repetition matters, fixed length</i> <i>Equal n-tuples have corresponding entries equal</i>	
string ordered finite sequence of elements each from specified set <i>repetition matters, arbitrary finite length</i> <i>Equal strings have same length and corresponding characters equal</i>	

Special cases:

When $n = 2$, the 2-tuple is called an **ordered pair**.

A string of length 0 is called the **empty string** and is denoted λ .

A set with no elements is called the **empty set** and is denoted $\{\}$ or \emptyset .

Sets

A set is an unordered collection of *elements*.

Recall that the empty set (denoted \emptyset or $\{\}$) is the set without any elements.

To define sets:

To define a set using **roster method**, explicitly list its elements using *curly braces*. That is, start with $\{$ then list elements of the set separated by commas and close with $\}$.

To define a set using **set builder definition**, either form “The set of all x from the universe U such that x is ...” by writing

$$\{x \in U \mid \dots x \dots\}$$

or form “the collection of all outputs of some operation when the input ranges over the universe U ” by writing

$$\{\dots x \dots \mid x \in U\}$$

We use the symbol \in as “is an element of” to indicate membership in a set.

Example sets: For each of the following, identify whether it's defined using the roster method or set builder notation and give an example element.

$$\{-1, 1\}$$

$$\{0, 0\}$$

$$\{-1, 0, 1\}$$

$$\{(x, x, x) \mid x \in \{-1, 0, 1\}\}$$

$$\{\}$$

$$\{x \in \mathbb{Z} \mid x \geq 0\}$$

$$\{x \in \mathbb{Z} \mid x > 0\}$$

$$\{\mathbf{A}, \mathbf{C}, \mathbf{U}, \mathbf{G}\}$$

$$\{\mathbf{AUG}, \mathbf{UAG}, \mathbf{UGA}, \mathbf{UAA}\}$$

***n*-tuples**

An *n*-tuple is a sequences of *n* *entries*.

To define n-tuples

To define *n*-tuples, list the entries separated by commas and enclosed by parentheses.

Example *n*-tuples: For each of the following *n*-tuple, come up with where it might come from. For example, (88, 79, 100, 90, 68, 81, 90) may be the grades a particular student will get for homeworks 1-7.

$$(1, 2.22, -0.8)$$

$$(-1, -1, 0, 1, 0, -1, 0, 0)$$

$$(CSE20, CSE21, CSE101, CSE105)$$

$$(MILES, JONES, A000000, MUIR, 91.86, A-)$$

strings (over a finite alphabet)

A string over a finite alphabet is a finite sequence of *characters* or *symbols*.

Recall that the *empty string*, (usually denoted λ) is the string without any characters.

The characters you are allowed to use come from a finite set called an *alphabet*.

To define strings:

Usually, a string is given as a list of characters without any punctuation. In some cases, we will use the symbol “ \circ ” as a delimiter. For example, if the alphabet is $\{\mathbf{A}, \mathbf{C}, \mathbf{U}, \mathbf{G}\}$, then consider the string **AAUA**. It could also be written as $\mathbf{A} \circ \mathbf{A} \circ \mathbf{A} \circ \mathbf{U} \circ \mathbf{A}$.

string concatenation

String concatenation of strings x and y , denoted xy or $x \circ y$ is the result of “gluing together” the strings x first then y . For example: **AUAA** \circ **GGC** = **AUAAGGC**.

Pro-tip: the meaning of writing one element next to another like xy depends on the data-types of x and y . When x and y are strings, the convention is that xy is the result of string concatenation. When x and y are numbers, the convention is that xy is the result of multiplication. This is (one of the many reasons) why is it very important to declare the data-type of variables before we use them.

Example strings: For each example, come up with an appropriate alphabet:

MATHEMATICS

A00000000

2598960

CCACUGCCA

λ

9500 *Gilman Dr.*

11100010101101000111111011111

mej016@eng.ucsd.edu

F96A4B0B

RNA is made up of strands of four different bases that encode genomic information in specific ways. The bases are elements of the set $B = \{\mathbf{A}, \mathbf{C}, \mathbf{U}, \mathbf{G}\}$.

Formally, to define the set of all RNA strands, we need more than roster method or set builder descriptions.

New! Recursive Definitions of Sets: The set S (pick a name) is defined by:

Basis Step:	Specify finitely many elements of S
Recursive Step:	Give rule(s) for creating a new element of S from known values existing in S , and potentially other values.

The set S then consists of all and only elements that are put in S by finitely many (a nonnegative integer number) of applications of the recursive step after the basis step.

Definition The set of nonnegative integers \mathbb{N} is defined (recursively) by:

Basis Step:
Recursive Step:

Examples:

Definition The set of all integers \mathbb{Z} is defined (recursively) by:

Basis Step:
Recursive Step:

Examples:

Definition The set of RNA strands S is defined (recursively) by:

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

where sb is string concatenation.

Examples:

Definition The set of bitstrings (strings of 0s and 1s) is defined (recursively) by:

Basis Step:
Recursive Step:

Notation: We call the set of bitstrings $\{0, 1\}^*$.

Examples:

To define a set we can use the roster method, set builder notation, a recursive definition, and also we can apply a set operation to other sets.

New! Cartesian product of sets and set-wise concatenation of sets of strings

Definition: Let X and Y be sets. The **Cartesian product** of X and Y , denoted $X \times Y$, is the set of all ordered pairs (x, y) where $x \in X$ and $y \in Y$

$$X \times Y = \{(x, y) \mid x \in X \text{ and } y \in Y\}$$

Definition: Let X and Y be sets of strings over the same alphabet. The **set-wise concatenation** of X and Y , denoted $X \circ Y$, is the set of all results of string concatenation xy where $x \in X$ and $y \in Y$

$$X \circ Y = \{xy \mid x \in X \text{ and } y \in Y\}$$

Pro-tip: the meaning of writing one element next to another like xy depends on the data-types of x and y . When x and y are strings, the convention is that xy is the result of string concatenation. When x and y are numbers, the convention is that xy is the result of multiplication. This is (one of the many reasons) why is it very important to declare the data-type of variables before we use them.

Fill in the missing entries in the table:

Set	Example elements in this set:			
B	A	C	G	U
	(A, C)	(U, U)		
$B \times \{-1, 0, 1\}$				
$\{-1, 0, 1\} \times B$				
			(0, 0, 0)	
$\{A, C, G, U\} \circ \{A, C, G, U\}$				
			GGGG	

New! Defining functions A function is defined by its (1) domain, (2) codomain, and (3) rule assigning each element in the domain exactly one element in the codomain.

The domain and codomain are nonempty sets.

The rule can be depicted as a table, formula, or English description.

The notation is

“Let the function $\text{FUNCTION-NAME: DOMAIN} \rightarrow \text{CODOMAIN}$ be given by
 $\text{FUNCTION-NAME}(x) = \dots$ for every $x \in \text{DOMAIN}$ ”.

or

“Consider the function $\text{FUNCTION-NAME: DOMAIN} \rightarrow \text{CODOMAIN}$ given by
 $\text{FUNCTION-NAME}(x) = \dots$ for every $x \in \text{DOMAIN}$ ”.

Example: The absolute value function

Domain

Codomain

Rule

Recall our representation of Netflix users' ratings of movies as n -tuples, where n is the number of movies in the database. Each component of the n -tuple is -1 (didn't like the movie), 0 (neutral rating or didn't watch the movie), or 1 (liked the movie).

Consider the ratings $P_1 = (-1, 0, 1)$, $P_2 = (1, 1, -1)$, $P_3 = (1, 1, 1)$, $P_4 = (0, -1, 1)$

Which of P_1 , P_2 , P_3 has movie preferences most similar to P_4 ?

One approach to answer this question: use **functions** to define distance between user preferences.

For example, consider the function d_0 :
given by

→

$$d_0((x_1, x_2, x_3), (y_1, y_2, y_3)) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$

Extra example: A new movie is released, and P_1 and P_2 watch it before P_3 , and give it ratings; P_1 gives ✓ and P_2 gives ✗. Should this movie be recommended to P_3 ? Why or why not?

Extra example: Define a new function that could be used to compare the 4-tuples of ratings encoding movie preferences now that there are four movies in the database.

When the domain of a function is a *recursively defined set*, the rule assigning images to domain elements (outputs) can also be defined recursively.

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

$$\begin{array}{ll} \text{Basis Step:} & \mathbf{A} \in S, \mathbf{C} \in S, \mathbf{U} \in S, \mathbf{G} \in S \\ \text{Recursive Step:} & \text{If } s \in S \text{ and } b \in B, \text{ then } sb \in S \end{array}$$

where sb is string concatenation.

Definition (Of a function, recursively) A function $rnalen$ that computes the length of RNA strands in S is defined by:

$$\begin{array}{lll} & & rnalen : S \rightarrow \mathbb{Z}^+ \\ \text{Basis Step:} & \text{If } b \in B \text{ then} & rnalen(b) = 1 \\ \text{Recursive Step:} & \text{If } s \in S \text{ and } b \in B, \text{ then} & rnalen(sb) = 1 + rnalen(s) \end{array}$$

The domain of $rnalen$ is

The codomain of $rnalen$ is

Example function application:

$$rnalen(\mathbf{ACU}) =$$

Extra example: A function $basecount$ that computes the number of a given base b appearing in a RNA strand s is defined recursively:

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

$$basecount((\mathbf{ACU}, \mathbf{A})) = basecount((\mathbf{AC}, \mathbf{A})) = basecount((\mathbf{A}, \mathbf{A})) = 1$$

$$basecount((\mathbf{ACU}, \mathbf{G})) = basecount((\mathbf{AC}, \mathbf{G})) = basecount((\mathbf{A}, \mathbf{G})) = 0$$

Extra example: The function which outputs 2^n when given a nonnegative integer n can be defined recursively, because its domain is the set of nonnegative integers.