

Digression: Real Numbers

- \mathbf{N} : the set of natural numbers (inductively defined)
- $f: \mathbf{N} \rightarrow \{0, 1, \dots, 9\}$ and $d \in \mathbf{N}$ (for exponent, decimal position)
- Define a real number as $x = (f, d)$, e.g.:
 - $\{(i, 9) \mid i \in \mathbf{N}\}, 0 \Leftrightarrow 0.999\dots$
 - $\{(0, 0)\} \cup \{(i, 9) \mid i \in \mathbf{N}, i > 0\}, 1 \Leftrightarrow 0.999\dots$
 - $\{(0, 1)\} \cup \{(i, 0) \mid i \in \mathbf{N}, i > 0\}, 1 \Leftrightarrow 1.000\dots$
 - $\{(0, 1), (1, 2), (2, 3)\} \cup \{(i, 0) \mid i \in \mathbf{N}, i > 2\}, 3 \Leftrightarrow 123.000\dots$
- \mathbf{R} : the set of all x 's How many such numbers?
 - No arithmetic operations can distinguish $1.000\dots$ and $0.999\dots$
E.g., $1.000\dots - 0.999\dots = 0.000\dots$, $1.000\dots / 3 = 0.999\dots / 3$
 - Partition induced by '=': $\{\dots, [0.999\dots], \dots\}$
 - $[0.999\dots] = [1.000\dots] = \{0.999\dots, 1.000\dots\}$ $|\mathbf{R}| = |\mathcal{P}(\mathbf{N})| > |\mathbf{N}|$

CMSC210 D3 1

D2: Correction

Lemma 2: R is symmetric.

Def: For any x, y , if $(x, y) \in R$, then $(y, x) \in R$

Hypotheses

- m is a positive integer.
- $R = \{(x, y) \mid (x \bmod m) = (y \bmod m)\}$
- $(x, y) \in R$

Conclusion: $(y, x) \in R$

1. $(x \bmod m) = (y \bmod m)$ [Hyp.]
2. $(y \bmod m) = (x \bmod m)$ [symmetric '=']
3. R is symmetric. [Def. symmetric]

CMSC210 D3 2

Review

Properties of Relations/Functions

Check properties *only for applicable cases*

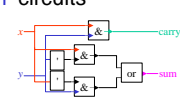
- Relation $_ \times _$
 - $A \times A \Rightarrow$ Reflexivity, symmetry, transitivity
- Function $_ \rightarrow _$
 - Closedness, surjectivity, injectivity, bijectivity
 - $A \times A \rightarrow A \Rightarrow$ associativity, existence of an identity element, existence of inverses, commutativity
 - If negation also available \Rightarrow De Morgan
 - If two functions available \Rightarrow Distributivity

CMSC210 D3 3

Intro

Minimalist Approaches

- Computer design
 - Using only 'or' and 'negation' circuits
 - Step 1: Build 'adders'
 - Step 2: Build 'multipliers'
 - Step 3: Build 'remainder'
- Arithmetic
 - Using only 0 and the successor function
 - Define 'addition', 'multiplication', 'remainder'



Other examples?

CMSC210 D3 4

Unit D3: Functions

Today

- Understand and use recursive function definition on numbers
- Explore more variations of recursive function definition
- Prove/disprove properties of functions
- Take-home exercises
 - Propositional logic, String operations [optional]

CMSC210 D3 5

Section 1

Building Blocks

$\mathbf{Nat} = (\mathbf{N}, succ, 0)$

- \mathbf{N} : the set of natural numbers
- $succ: \mathbf{N} \rightarrow \mathbf{N}$ abbreviation: $succ(x)$ as " $x + 1$ "
 - E.g., $succ(0) = 1, succ(1) = 2$
- $0: \rightarrow \mathbf{N}$
- This structure (including its components) can be defined/specified inductively [D1].

CMSC210 D3 6

Addition

- A definition based on 0 and *succ* (and this function itself) abbreviation: $add(x, y)$ as " $x + y$ "

$$add(x, y) = \begin{cases} y & \text{if } x = 0 \\ add(z, succ(y)) & \text{otherwise} \\ \text{where } succ(z) = x & \end{cases}$$

- Using '- 1' as shorthand Try $add(2, 3)$

$$add(x, y) = \begin{cases} y & \text{if } x = 0 \\ add(x - 1, succ(y)) & \text{otherwise} \end{cases}$$

CMSC210 D3

7

Recursive Function Definition

$$add(x, y) = \begin{cases} y & \text{if } x = 0 \leftarrow \text{"base case"/"basis"} \\ add(x-1, succ(y)) & \text{otherwise} \leftarrow \text{"induction (step)"} \end{cases}$$

- Recursive definition: **function vs. set**
 - Both have the base case and induction step
 - Exclusion clause: **only for set definition** Why?

Another comparison point

- Set: Grow the set
- Function: Reach the base case

CMSC210 D3

8

Programming Languages

- Recursion
 - General way of representing everything computable, including iteration
- Iteration (loop)
 - Special case of recursion

CMSC210 D3

9

Practice: Addition (Alternatives)

- Which is the correct alternative definition?

$$add_1(x, y) = \begin{cases} y & \text{if } x = 0 \\ add_1(succ(x-1), y) & \text{otherwise} \end{cases}$$

$$add_2(x, y) = \begin{cases} y & \text{if } x = 0 \\ succ(add_2(x-1), y) & \text{otherwise} \end{cases}$$

Try $add(2, 3)$

CMSC210 D3

10

Group Exercise 1: Multiplication

- Recursively define multiplication based on 0, *succ*, and *add* (and this function itself) May abbreviate $mult(x, y)$ as " $x \times y$ "

$$mult(x, y) = \begin{cases} \end{cases}$$

CMSC210 D3

11

Practice: Factorial

- A definition based on the previously-defined functions (and this function itself) abbreviation: $factorial(x)$ as " $x!$ "

$$x! = \begin{cases} \end{cases}$$

Can we define any function this way?

CMSC210 D3

12

Practice: '<'

- A definition based on the previously-defined functions (and this function itself)
- Note: Assume some fixed c

$$x < c = \left\{ \begin{array}{l} \end{array} \right.$$

CMSC210 D3

13

Practice: Remainder

- A definition based on the previously-defined functions (and this function itself)
- Note: Assume some fixed k

$$x \bmod k = \left\{ \begin{array}{l} \end{array} \right.$$

CMSC210 D3

14

Group Exercise 2: Permutation

- Recursively define permutation based on the previously-defined functions (and this function itself)

$$P(n, r) = n! / (n - r)! \\ = n \times (n - 1) \times \dots \times (n - r + 1)$$

Try $P(5, 3)$

Hint: How to represent: $(n - 1) \times \dots \times (n - r + 1)$?

CMSC210 D3

15

Section Summary

- Recursive function definition is a means to define any function (numeric or **non-numeric**).
- Recursive definition can represent **potentially infinite** phenomena in a finite manner.
- Recursive function definition
 - Base + Induction step, **NO** exclusion clause
 - Computation will stop at the base case.

CMSC210 D3

16

Section 2

Group Exercise 3: String Length

Strings = $(\Sigma, \Sigma^*, \text{attachChar}, e)$

- Σ : the set of printable characters
- Σ^* : the set of all strings
- $\text{attachChar}: \Sigma^* \times \Sigma \rightarrow \Sigma^*$
- $e: \rightarrow \Sigma^*$

- **N**: the set of natural numbers (including 0)
- Recursively define a function **length** of the type $\Sigma^* \rightarrow \mathbf{N}$ that would compute the length of a string.

May use previously-defined functions

CMSC210 D3

17

Practice: Producer-Consumer

- Condition for termination?

$$\text{produce}(m) = \begin{cases} m & \text{if } m > 9 \\ \text{consume}(m \times 1.25) & \text{otherwise} \end{cases}$$
$$\text{consume}(m) = \begin{cases} m & \text{if } m < 1 \\ \text{produce}(m \times 0.75) & \text{otherwise} \end{cases}$$

Mutual recursion

CMSC210 D3

18

Section Summary

- Recursive function definitions
 - On non-numeric objects
 - Depending on other recursive functions
- The notion of “[computation](#)” can be completely formalized with recursive functions.

Summary Exercise

- Describe your readiness for today’s take-home exercises
 - Propositional logic: check wff, find the truth value
 - String operations: e.g., replace substring
- [Questions/Comments/Suggestions](#)