CS 242

# Computable Functions

John Mitchell

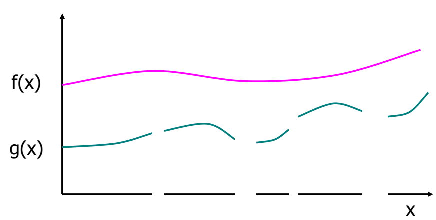
Reading: Chapter 2

## Foundations: Partial,Total Functions

- Value of an expression may be undefined
  - Undefined operation, e.g., division by zero
    - 3/0 has no value
    - implementation may halt with error condition
  - Nontermination
    - f(x) = if x=0 then 1 else f(x-2)
    - this is a *partial* function: not defined on all arguments
    - cannot be detected at compile-time; this is halting problem
  - These two cases are
    - "Mathematically" equivalent
    - Operationally different

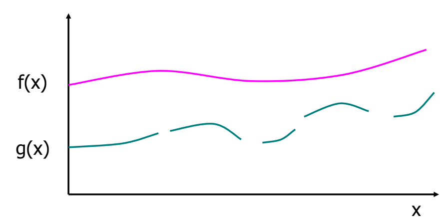
Subtle: "undefined" is not the name of a function value …

## Partial and Total Functions




- Total function: f(x) has a value for every x
- Partial function: g(x) does not have a value for every x

## Functions and Graphs




- Graph of $f = \{ \langle x,y \rangle \mid y = f(x) \}$
- Graph of $g = \{ \langle x,y \rangle \mid y = g(x) \}$

Mathematics: a function is a set of ordered pairs (graph of function)

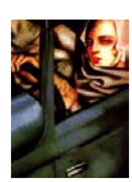
## Partial and Total Functions

- Total function $f: A \to B$ is a subset $f \subseteq A \times B$ with
  - For every $x \in A$, there is some $y \in B$ with $\langle x,y \rangle \in f$ (total)
  - If $\langle x,y \rangle \in f$ and $\langle x,z \rangle \in f$ then $y=z$ (single-valued)
- Partial function $f: A \to B$ is a subset $f \subseteq A \times B$ with
  - If $\langle x,y \rangle \in f$ and $\langle x,z \rangle \in f$ then $y=z$ (single-valued)
- Programs define partial functions for two reasons
  - partial operations (like division)
  - nontermination

    f(x) = if x=0 then 1 else f(x-2)

## Halting Problem

Entore Buggati: "I build cars to go, not to stop."

Self-Portrait in the Green Buggati (1925)
Tamara DeLempicka

## Computability

- Definition
  Function f is computable if some program P computes it:
  For any input x, the computation P(x) halts with output f(x)

- Terminology
  Partial recursive functions
  = partial functions (int to int) that are computable

## Halting function

- Decide whether program halts on input
  - Given program P and input x to P,

$$Halt(P,x) = \begin{cases} \text{yes} & \text{if } P(x) \text{ halts} \\ \text{no} & \text{otherwise} \end{cases}$$

Clarifications
Assume program P requires one string input x
Write P(x) for output of P when run in input x
Program P is string input to *Halt*

Fact: There is no program for *Halt*

## Unsolvability of the halting problem

- Suppose P solves variant of halting problem
  - On input Q, assume

$$P(Q) = \begin{cases} \text{yes} & \text{if } Q(Q) \text{ halts} \\ \text{no} & \text{otherwise} \end{cases}$$

- Build program D
  - $D(Q) = \begin{cases} \text{run forever} & \text{if } Q(Q) \text{ halts} \\ \text{halt} & \text{if } Q(Q) \text{ runs forever} \end{cases}$

- Does this make sense? What can D(D) do?
  - If D(D) halts, then D(D) runs forever.
  - If D(D) runs forever, then D(D) halts.
  - *CONTRADICTION:* program P must not exist.

## Main points about computability

- Some functions are computable, some are not
  - Halting problem

- Programming language implementation
  - *Can* report error if program result is undefined due to division by zero, other undefined basic operation
  - *Cannot* report error if program will not terminate

## Diversion: Theme Songs

- C "Iron Man," Black Sabbath
  " . . .Kills the people he once saved . . . "

- C++ "Imperial March (Darth Vader's Theme)," John Williams
  That'd be from "The Empire Strikes Back"

- Java "Goody Two Shoes," Adam Ant
  "Don't drink don't smoke - what do you do?"

- Perl "Oops! . . . I Did it Again," Britney Spears
  Feel free to substitute your favorite error-prone language.

Contributed by David Burden, HP Colorado