course: CSC 135-01 - Computing Theory and Programming Languages

instructor: Ted Krovetz

related notes: 2022-04-26

Computability - Study of what's computable

W17.2 | Tuesday, April 26, 2022 | 09:04 AM

Announcements

- No class 2022-04-28
- Exams back 2022-04-30

Turing Machines

Is a computable application/problem

Decision Problems

A problem with a `yes`/`no` answer

Most Famous Non-computable Problem: `Halt`

Given program **M** and input **x**, does $M(x)$ ever Terminate?

- $Halt\,(M,x)$: Not computable for **ALL** (M, x)
  - `true` if $M(x)$ eventually halts
  - `false`if $M(x)$ never halts (i.e. infinite loop)

Affirmative

Given **program M** and **Input x** does M(x) output "`True`"

- Affirmative(M,x)
  - `true` if M(x) outputs true
  - `false` outputs
    - M(x) outputs false

- M(x) loops

Strategy To Show Halt Not Computable

1. Show "Affirmative" not computable
2. Reduce from Affirmative to Halt

Reductions

Reductions

Problem **A** reduces Problem **B**
if a Problem **B** solver could be used as a subroutine in a Problem **A** solver

$$\exists\ \text{BSolver} \rightarrow \exists\ \text{ASolver}$$

Tldr

This doesn't mean that there is an existence of "BSolver" or "ASolver", but simply implies if "BSolver" is to be then so argues "ASolver". Proof via a the contrapositive...

$$\exists\ \text{BSolver} \rightarrow \exists\ \text{ASolver}$$

$$\neg \exists\ \text{ASolver} \rightarrow \neg \exists\ \text{BSolver}$$

Abstract

```
def Asolver(a_instance):
        b_instance = preprocess(a_instance)
        b_solution = Bsolver(b_instance)
```

```
        a_solution = postprocess(b_solution)
        return a_solution
```

Reduction Example01: Minimal Value in Array

Example

Finding the min in an array reduces to sorting an array

```
def min(arr)
        tmp = sort(arr)
        min_value = tmp[0]
        return min_value
```

Reduction Example02

Proves with step 2: Reduce from Affirmative to Halt

Given Affirmative not computable
Show via reduction that `Halt` is not computable

```
Affirmative(M, x):
        if Halt(M, x) == True:
                return M(x)
        else:
                return False
```

Theorem: Affirmative is Not Computable

Proof Sketch

Assume for contradiction Affirmative is computable

Define: `D(M)`:

> ```
> if Affirmative (M, M) == True:
>         return False
> else:
>         return True
> ```
>
> Consider what happens when you run `D(D)`
>
> ```
> if Affirmative(D, D) == True:
>         return False
> else:
>         return True
> ```

Case 1: If Affirmative(D, D) == True then

(D, D) is TRUE by definition of Affirmative,
But D(D) outputs FALSE by definition of D.
Both can't be TRUE....

Case 2: if Affirmative(D, D) == FALSE then

D(D) does not output TRUE by definition of Affirmative
But D(D) outputs TRUE by definition of D

In all cases: Contradiction