

## Computational Models

### 5. Decision Problems

## Decision Problems

- Input: Element from some (infinite) set of representations (e.g. strings)
- Output: Yes/No
- A problem is *decidable* if there is a (TM) program that always gives the right answer.
- A problem is *semi-decidable* if there is a (TM) program that gives the right answer whenever the answer is yes and never gives the wrong answer.

Computational Models

2

## Examples

- |  |                                       |
|--|---------------------------------------|
| • FSA emptiness<br>– $L(A) =? \emptyset$ | • FSA equivalence<br>– $L(A) =? L(B)$ |
| • CFG emptiness<br>– $L(G) =? \emptyset$ | • CFG equivalence<br>– $L(G) =? L(H)$ |
| • TM emptiness<br>– $L(M) =? \emptyset$  | • TM equivalence<br>– $L(M) =? L(N)$  |

Computational Models

3

## Examples

- |  |                                       |
|--|---------------------------------------|
| • FSA emptiness<br>– $L(A) =? \emptyset$ | • FSA equivalence<br>– $L(A) =? L(B)$ |
| • CFG emptiness<br>– $L(G) =? \emptyset$ | • CFG equivalence<br>– $L(G) =? L(H)$ |
| • TM emptiness<br>– $L(M) =? \emptyset$  | • TM equivalence<br>– $L(M) =? L(N)$  |

Computational Models

4

## Halting Problem

*Inputs:*

Program (for example, TM code)  $f$

Input value  $x$

*Question:*

Does the computation of  $f$  on  $x$  halt?

Computational Models

5

## $\text{Halt}_0$ Problem

*Input:*

Input-less program  $f$

*Question:*

Does the computation of  $f$  halt?

Computational Models

6

## $\text{Halt}^*$ (Universal Halting) Problem

*Input:*

Single-input program  $f$

*Question:*

Does the computation of  $f$  *always* halt?

Computational Models

7

## Undecidability of Halting

Consider the program:

```
c(f) := if halt(f,f) then loopy() else T
where loopy() := loopy()
      halt := ...
```

What is the result of executing  $c(c)$ ??!

Computational Models

8

## Halt Function

$\perp$  means “undefined”

$f(a) \equiv g(a)$  means  $f(a)$  and  $g(a)$  are both  $\perp$  or they are both defined and equal

$\text{halt}(f,x) = \text{if } f(x) = \perp \text{ then F else T}$   
 $\text{halt} = \lambda f,x (f(x) \neq \perp)$

Computational Models

9

## Proof

$c(c) = \text{if } \text{halt}(c,c) \text{ then } \text{loopy}() \text{ else T}$

- $\text{halt}(c,c)=F \Rightarrow c(c)=T \Rightarrow \text{halt}(c,c)=T$
- $\text{halt}(c,c)=T \Rightarrow c(c)=\perp \Rightarrow \text{halt}(c,c)=F$
- $\text{halt}(c,c)=\perp$

**In all 3 cases, halt doesn't work as advertised.**

Computational Models

10

## General (Turing) reductions

- Problem A *reduces* to problem B.
- If B is solvable, then so is A.
- If A is undecidable, then B is also not decidable.
- Notation:  $A \propto B$

Computational Models

11

## Correctness of $A \propto B$

- Assuming that B is decidable, i.e. we have a terminating program for B, then...

–  $A(x)=T \Leftrightarrow x \in A$

–  $A(x)=F \Leftrightarrow x \notin A$

Computational Models

12

## Example of reduction

Reduction:  $\text{HALT} \propto \text{HALT}_0$

$\text{halt}(f,x) := \text{halt}_0(\lambda.f(x))$

Proof of correctness:

- (a) if  $f$  halts on  $x$  then  $\lambda.f(x)$  halts and  $\text{halt}_0$  answers T, which is correct;
- (b) if  $f$  does not halt on  $x$  then  $\lambda.f(x)$  never halts and  $\text{halt}_0$  answers F, which is correct

Computational Models

13

## Important Programs

- Check validity of a program  $\text{Good?}(f)$
- Generate program number  $n$   $\text{Prog}_n$
- Compute the result of  $f(x)$   $I(f,x)$

Computational Models

14

## Interpreters

An *interpreter* is a program  $I$  that takes as input a pair  $(f,x)$  consisting of a program  $f$  and input  $x$  and computes  $f(x)$ .

In *Scheme* you have interpretation for free:

$(I f x1 \dots xn) := (f x1 \dots xn)$

Computational Models

15

## Universality

An interpreter serves as a *universal* program in that it is the only program needed to compute all functions computable in the given model:

To compute  $f(x)$ , run  $I(f,x)$ .

The language in which the interpreter  $I$  is written need not be the same as the language in which programs  $f$  are written:

$I(\underline{f},x) = \underline{f}(x)$

Computational Models

16

## Steppers

A *stepper* is a program that computes

$SP(f,x,n)$  = if “ $f(x)$  halts within bound  $n$ ”  
then  $f(x)$   
else “fail”

Computational Models

17

## Stepper $\Rightarrow$ Interpreter

$I(f,x) := I'(f,x,0)$

where

$I'(f,x,n) :=$

if  $SP(f,x,n) = \text{fail}$

then  $I'(f,x,n+1)$

else  $SP(f,x,n)$

Computational Models

18