Undecidability

Section 4.2

Algorithmically Unsolvable Problems

- Many problems are unsolvable by computers
  - Many tasks that seem simple, may be computationally impossible
- Previously, we have used TM to show that a problem is solvable
  - Encode a problem as a language
  - If a TM is created that can decide the language, it is solvable.
- Now we introduce techniques to show that a problem is unsolvable.

An Undecidable Problem

- Problem: Is it possible to determine whether a Turing Machine accepts a given input string?
- Formulate this problem as a language A_{TM} = {<M,w>|M is a TM and M accepts w}.
- This language is recognizable by creating a TM that simulates M
  - U = "On input <M,w>, where M is a TM and w is a string:
    - 1. Simulate M on input w
    - 2. If M enters an accept state, accept. If it enters a reject state, reject.
  - U loops if M loops
    - Not guaranteed to halt, therefore is not a decider
- U is a Universal Turing Machine
  - A TM that is capable of simulating any other TM

Correspondence

- In order to show that not every problem is computable
  - Assume that for every unique problem, a unique TM must be created to solve it
  - This means the set of all problems, S_p, must be the same size of the set of all Turing Machines, S_{TM}
- Both sets are infinite but one may be larger than the other
  - For sets to be the same size, there must be a correspondence (bijective) between every element in each set.
    - One-to-one (injective)
    - Onto (surjective)

Countable Set

- If a set has a correspondence to the set of natural numbers N, that set is said to be countable
- If we cannot find a correspondence to N, then the set is uncountable
  - Uncountable sets are larger than countable sets.

Example: Countable Set

- Show that the set of even numbers, E, is countable

- Show correspondence between E and N
  - f(n) = 2n

- E is countable

n	f(n)
1	2
2	4
3	6
…	…

Example: Uncountable Set

- Show that the set of real numbers, R, is uncountable

- Procedure
  - Systematically construct a list for R
    - The index of each element in the list corresponds to an element in N
  - Find an element, x, in R that is cannot be the list
    - Diagonalization method
      - Choose the digits for x so that x ≠ f(n) for any n

- For the following list, choose a number that is different from the diagonal
  - Uses each digit to mismatch the corresponding element
  - For x ≠ f(1), 1st digit must be different
  - For x ≠ f(n), nth digit must be different

- x = 0.4641…

n	f(n)
1	3.<u>1</u>4159…
2	55.5<u>5</u>555…
3	0.12<u>3</u>45…
4	0.500<u>0</u>0…
…	…

Uncountable Number of Languages

- Since there are problems related to real numbers, the set of all problems S_p is uncountable
- The set of all TMs S_{TM} can be listed and is countable
- This means is S_p larger than S_{TM}
  - and that there are problems without a corresponding TM

Example: An Undecidable Language

- Problem: Is it possible to determine whether a TM accepts a given input string?

- Formulate this problem as a language A_{TM} = {<M,w>|M is a TM and M accepts w}.
  - Assume that A_{TM} is decidable and obtain a contradiction
  - Create a decider H for A_{TM} :

$$H(< M, w >) = \begin{cases} accept & if\ M\ accepts\ w \\ reject & if\ M\ does\ not\ accept\ w \end{cases}$$

- Create a TM D with H as a subroutine
  - D = “On input <M>, where M is a TM:
    1. Run H on input <M,<M>>.
    2. Output the opposite of what H outputs. That is, if H accepts, reject; and if H rejects, accept.”

Example: An Undecidable Language

- Is it possible to determine whether a Turing Machine accepts a given input string?
- Formulate this problem as a language A_{TM} = {<M,w>|M is a TM and M accepts w}.
  - Assume that A_{TM} is decidable and obtain a contradiction
  - Create a decider H for A_{TM} :

$$H(< M,w >) = \begin{cases} accept & if\ M\ accepts\ w \\ reject & if\ M\ does\ not\ accept\ w \end{cases}$$

- Create a TM D to <u>simulate diagonalization</u> with H as a subroutine.
  - D does the <u>opposite</u> what M does when it receives itself as an input
  - D = “On input <M>, where M is a TM:
    1. Run H on input <M,<M>>.
    2. Output the opposite of what H outputs. That is, if H accepts, reject; and if H rejects, accept.”

Example: An Undecidable Language

- D always does the opposite

$$D(< M >) = \begin{cases} accept & if\ M\ does\ not\ accept < M > \\ reject & if\ M\ accepts < M > \end{cases}$$

- If D receives itself, then

$$D(< D >) = \begin{cases} accept & if\ D\ does\ not\ accept < D > \\ reject & if\ D\ accepts < D > \end{cases}$$

  - This is a contradiction, and we can use diagonalization to see this

Example: An Undecidable Language

- Create a table of TMs and their encode versions:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$		$\vdots$			

- Output of H:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*	
M_4	*accept*	*accept*	*reject*	*reject*	
$\vdots$		$\vdots$			

- Since D itself is a TM it will be on the list and will be the opposite of the diagonals
  - When we reach (D,<D>), we get a contradiction
- This means that such a TM does not exist
  - A_{TM} is undecidable

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*		*accept*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*		*reject*	
M_4	*accept*	*accept*	*reject*	*reject*		*accept*	
$\vdots$		$\vdots$			$\ddots$		
D	*reject*	*reject*	*accept*	*accept*		?	
$\vdots$		$\vdots$					$\ddots$