# Computability

## The Turing Machine

- Motivating idea
  - Build a theoretical a "human computer"
  - Likened to a human with a paper and pencil that can solve problems in an algorithmic way
  - The theoretical machine provides a means to determine:
    - If an algorithm or procedure exists for a given problem
    - What that algorithm or procedure looks like
    - How long would it take to run this algorithm or procedure.

## Theory Hall of Fame

- Alan Turing
  - 1912 – 1954
  - b. London, England.
  - PhD – Princeton (1938)
  - Research
    - Cambridge and Manchester U.
    - National Physical Lab, UK
  - Creator of the Turing Test

## The Church-Turing Thesis (1936)

- Any algorithmic procedure that can be carried out by a human or group of humans can be carried out by some Turing Machine"
  - Equating algorithm with running on a TM
  - Turing Machine is still a valid computational model for most modern computers.

## Theory Hall of Fame

- Alonso Church
  - 1903 -- 1995
  - b. **Washington D.C.**
  - PhD – Princeton (1927)
  - Mathematics Prof (1927 – 1967)
  - Advisor to both Turing and Kleene

## Undecidability

- Informally, a problem is called unsolvable or undecidable if there no algorithm exists that solves the problem.
- Algorithm
  - Implies a TM that computes a solution for the problem
- Solves
  - Implies will always give an answer

## Decision Problem

- Let's formalize this a bit
  - A decision problem is a problem that has a yes/no answer
  - Example:
    - Is a given string x a palindrome (Is x ∈ pal?)
    - Is a given context free language empty?

## Decision Problem

- Running a decision problem on a TM.
  - The problem must first be encoded
  - Example:
    - Is a given string x a palindrome (Is x ∈ pal?)
      - x is an instance of the problem
    - Is a given context free language empty?
      - Instance of a problem is a CFG…must be encoded.

## Decision Problem

- Running a decision problem on a TM.
  - Once encoded, the encoded instance in provided as input to a TM.
  - The TM must then
    - Determine if the input is a valid encoding
    - Run, halt,
      - Place 1 on the tape if the answer for the input is yes
      - Place 0 on the tape if the answer for the input is no
  - If such a TM exists for a given decision problem, the problem is decidable or solvable. Otherwise the problem is called undecidable or unsolvable.

## Solvability

- In other words, a problem is solvable if the language of all of its encoded "yes" instances is recursive.
  - There is a TM that recognizes the language.

## Universal Language

- Universal Language ($L_u$)
  - Set of all strings $w_i$ such that $w_i \in L(M_i)$
  - All strings w that are accepted by the TM with w as it's encoding.
  - All encodings for TMs that do accept their encoding when input
- We showed that $L_u$ is not recursive.

## An unsolvable problem

- $L_u$ corresponds to the "yes encodings" of the decision problem:
  - Given a Turing Machine M, does it accept it's own encoding. (Self-accepting)
- Since $L_u$ is not recursive, this problem is unsolvable.

## Reducing one language to another

- One method of showing whether a given decision problem is unsolvable is to convert the encoding of the problem into another that we know to be either solvable or unsolvable.
- This is called reducing one language to another.

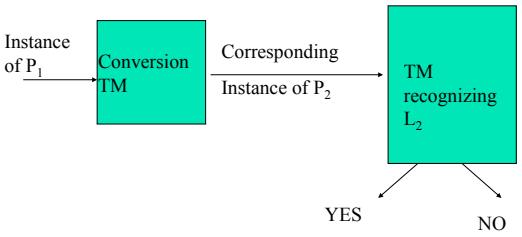
## Reducing one language to another

- Formally,
  - Let $L_1$ and $L_2$ be languages over $\Sigma_1$ and $\Sigma_2$
  - We say $L_1$ is reducible to $L_2$ if
    - There exists a Turning computable function
    - f: $\Sigma_1^* \rightarrow \Sigma_2^*$ such that
    - $x \in L_1$ iff $f(x) \in L_2$

## Reducing one language to another

- Informally,
  - We can take any encoded instance of one problem
    - Use a TM to compute a corresponding encoded instance of another problem.
    - If this other problem has a TM that recognizes the set of "yes encodings", we can run that TM to solve the first problem.

## Reducing one language to another




## Reducing one language to another

- Key facts:
  - If $L_1$ is reducible to $L_2$ then
    - If $L_2$ is recursive then $L_1$ is also recursive
    - If $L_1$ is not recursive then $L_2$ is not recursive.
  - If $P_1$ and $P_2$ are decision problems with $L_1$ and $L_2$ the languages of "yes encodings" respectively and if $L_1$ is reducible to $L_2$ then
    - If $P_2$ is solvable then $P_1$ is also solvable
    - If $P_1$ is unsolvable then $P_2$ is also unsolvable

## The halting problem

- Let's consider a more general problem about TMs.
  - Given a TM, M, and a string w, is $w \in T(M)$?
  - Given a TM, M and a string w
    - Will M halt and accept on input w?
  - We simply cannot just run the string on the TM since if $w \notin L(M)$, M might go into an infinite loop.

## The halting problem

- The halting problem is unsolvable
- Proof:
  - We can use an argument similar to that used to show that $L_u$ is not recursive.
  - Instead, let's use reduction

## To show a problem is unsolvable

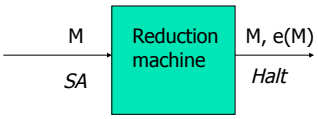
- Find a problem known to be unsolvable
- Reduce this known unsolvable problem to the problem you wish to show is unsolvable.
- Only need one to start the ball rolling
  - Self-accepting fits the bill.

## Halting problem

- Reduce self-accepting to halting
  - Self-accepting
    - Turing Machine M
    - Does this Machine accept it's own encoding
  - Halting
    - Turing Machine M
    - String w
    - Does M halt and accept on input w

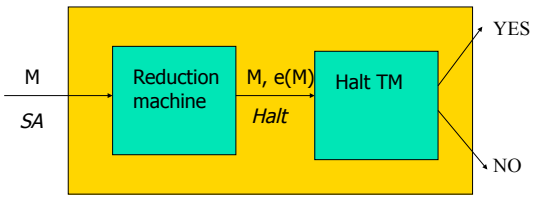
## Halting problem

- Reduction
  - Take an instance of SA and convert it to an instance of Halting
    - Such that a "yes" instance of SA results in a "yes" instance of halt




## Halting problem

- Reduction
  - Assume Halting is solvable




Then SA must be solvable…CONTRADICTION!

## State entry problem

- Given:
  - Turing Machine M
  - A state q
  - A string w
- Problem:
  - Will M enter state q on input w.
- The State Entry Problem is unsolvable.

## State entry problem

- Reduce halting to state entry (SE).
  - Halting
    - Turing Machine M
    - String w
    - Does M halt and accept on input w
  - State Entry
    - Turing Machine M
    - String w
    - State q
    - Does this Machine enter state q on input w.

## State entry problem

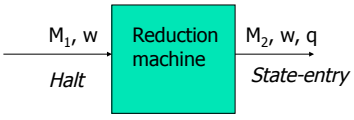
- Reduction
  - Take an instance of Halting ($M_1$, $w_1$) and convert it to an instance of state-entry ($M_2$, $w_2$, q)
    - Such that a "yes" instance of halt results in a "yes" instance of self-entry
    - From $M_1$ create $M_2$ such that $M_1$ halts iff $M_2$ enters state q.

## State entry problem

- Reduction
  - $M_1$ will halt only there is no transition defined (e.g. $\delta$ ($q_i$, a) )
  - Take $M_1$, create a new state q.
  - Define new transition in $M_2$ for each undefined transition in $M_1$ so
    - ($q_i$, a) = (q, a R)
  - If $M_1$ halts, $M_2$ will enter state q

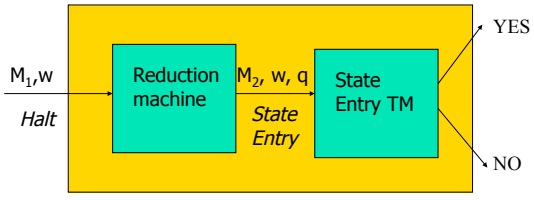
## State entry problem

- Reduction
  - Take an instance of Halting and convert it to an instance of state entry




## State entry problem

- Reduction
  - Assume State Entry is solvable




Then Halting must be solvable…CONTRADICTION!

## Strings of same length problem

- Given:
  - Turing Machine M
- Problem:
  - Will M enter accept two strings of the same length.
- The Same Length Problem is unsolvable.

## Same length problem

- Reduce halting to same length.
  - Halting
    - Turing Machine M
    - String w
    - Does M halt and accept on input w
  - State Entry
    - Turing Machine M
    - Does this Machine accept 2 strings of the same length.

## Same length problem

- Reduction
  - Take an instance of Halting ($M_1$, $w_1$) and convert it to an instance of state-entry ($M_w$)
    - Such that a "yes" instance of halt results in a "yes" instance of same length
    - From $M_1$ create $M_w$ such that $M_1$ halts on w iff $M_w$ accepts string of same length

## Same length problem

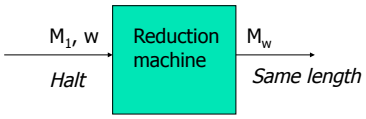
- Reduction
  - $M_1$ will halt only there is no transition defined (e.g. $\delta$ ($q_i$, a) )
  - Define new transition in $M_w$ for each undefined transition in $M_1$ so
    - ($q_i$, a) will force $M_2$ to accept a and b
  - If $M_1$ halts on w, $M_w$ accept "a" and "b" (strings of same length)

## Same length problem

- ($q_i$, a) will force $M_w$ to accept "a" and "b"
  - We can certainly create TMs that accept "a" and "b".
  - $M_w$ will do the following:
    - Copy w onto it's tape (after the input)
    - Place the tape head at the start of w
    - Simulate $M_1$
    - For all halting configurations, transition back to the start of the input (where you can accept both "a" and "b").
    - You will only get back to the start of input of $M_w$ if $M_1$ halts on input w.

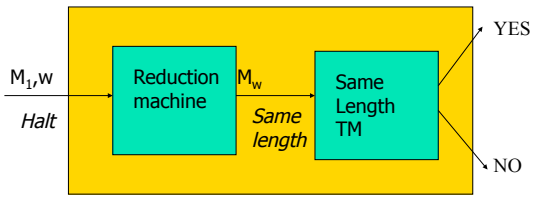
## Same length problem

- Reduction
  - Take an instance of Halting and convert it to an instance of same length




## Same length problem

- Reduction
  - Assume Same length is solvable




Then Halting must be solvable…CONTRADICTION!

## Decision Problems

- For recursively enumerable languages
  1. Is the language accepted by a TM empty?
  2. Is the language accepted by a TM finite?
  3. Is the language accepted by a TM regular?
  4. Is the language accepted by a TM context free?
  5. Is the language accepted by 1 TM a subset of or equal to the language accepted by another?

## Rice's Theorem

- Halting problem can be reduced to each one of these decision problems.
  - Using same argument as same length.
- Rice's Theorem
  - Every non-trivial property of recursively enumerable languages is unsolvable.
    - Where a non-trivial property is a property satisfied by any non-null subset of the set of recursively enumerable languages.

## Decision Problems

- For recursively enumerable languages
- All unsolvable.
  1. Is the language accepted by a TM empty?
  2. Is the language accepted by a TM finite?
  3. Is the language accepted by a TM regular?
  4. Is the language accepted by a TM context free?
  5. Is the language accepted by 1 TM a subset of or equal to the language accepted by another?

## Questions?

- Let's look at some more unsolvable problems…
- Some that don't have to do with recursively enumerable languages

## Post Correspondence Problem

- Given 2 lists of strings (each list with the same number of elements) can one pick a sequence of corresponding strings from the two lists and form the same string by concatenation. (PCP)
  - Attributed to Emil Post (1946).

## Theory Hall of Fame

- Emil Post
  - 1897 – 1954
  - b. Augustów, Poland.
  - PhD – Columbia (1920)
  - Research
    - Princeton.
    - Columbia
    - Cornell
  - Plagued by mental illness

## Post Correspondence Problem

- Example:

| | | | | | | |
|---|---|---|---|---|---|---|
| List 1 | 10 | 01 | 0 | 100 | 1 | 0 |
| List 2 | 101 | 100 | 10 | 0 | 010 | 00 |
| | 1 | 2 | 3 | 4 | 5 | 6 |

  - Choose a sequence of indicies : 1,3,4
- List1: 10 0 100  List 2: 101 10 0

## Post Correspondence Problem

- Is there a set of indices such that both lists produce the same string
  - Note: Indicies can be repeated

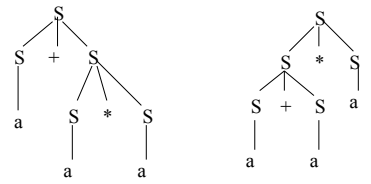
| | | | | | | |
|---|---|---|---|---|---|---|
| List 1 | 10 | 01 | 0 | 100 | 1 | 0 |
| List 2 | 101 | 100 | 10 | 0 | 010 | 00 |
| | 1 | 2 | 3 | 4 | 5 | 6 |

- Try 1, 4, 6
- List 1: 101000  List 2 :101000

## Post Correspondence Problem

- There is a Modified version of the Post Correspondence Problem (MPCP)
  - Requires that the index 1 appears as the first index in any solution.
  - This can be shown to be unsolvable by reducing the halting problem to MPCP
    - HALTING ≤ MPCP
  - MPCP can be reduced to PCP
    - HALTING ≤ MPCP ≤ PCP
  - Since halting is unsolvable
    - MPCP is unsolvable
    - PCP is unsolvable.

## Recall: Parse trees




Same string, 2 derivations

## CFG Ambiguity

- A CFG is said to be ambiguous if there is at least 1 string in L(G) having two or more distinct derivations.
- We said many weeks ago that there is no algorithm to determine if a given CFG is ambiguous.
  - Now we shall prove it

## CFG Ambiguity

- Given a CFG, the problem of whether this grammar is ambiguous is unsolvable.
  - Reduce PCP to Ambiguity.
  - Meaning:
    - Take an instance of PCP and convert it to a CGF G such that:
      - G is ambiguous iff the instance of PCP has a solution.

## CFG Ambiguity

- Instance of PCP
  - 2 Lists of strings A & B, all strings $\in \Sigma^*$
    - A = $(w_1, w_2, \ldots, w_n)$
    - B = $(x_1, x_2, \ldots, x_n)$
- Build a CFG, G with
  - Terminal set that includes $\Sigma$ plus special symbols $\{ a_1, a_2, \ldots a_n \}$ which represent indicies into lists A & B

## CFG Ambiguity

- Instance of PCP
  - 2 Lists of strings A & B, all strings $\in \Sigma^*$
    - A = $(w_1, w_2, \ldots, w_n)$
    - B = $(x_1, x_2, \ldots, x_n)$
- Productions of G
  - $A \rightarrow w_1Aa_1 \mid w_2Aa_2 \mid \ldots \mid w_nAa_n$
  - $A \rightarrow w_1a_1 \mid w_2a_2 \mid \ldots \mid w_na_n$
  - $B \rightarrow x_1Ba_1 \mid x_2Ba_2 \mid \ldots \mid x_nBa_n$
  - $B \rightarrow x_1a_1 \mid x_2a_2 \mid \ldots \mid x_na_n$
  - $S \rightarrow A \mid B$

## CFG Ambiguity

- Must show that G is ambiguous iff PCP instance has a solution.
  - Assume PCP has a solution (i1,i2, …,im)
  - Consider the derivations
    - $S \Rightarrow w_{i1} Aa_{i1} \Rightarrow w_{i1} w_{i2}Aa_{i2}a_{i1} \Rightarrow \ldots \Rightarrow$
    - $w_{i1} w_{i2} \ldots w_{im} A\, a_{im} \ldots a_{i2}a_{i1} \Rightarrow w_{i1} w_{i2} \ldots w_{im}a_{im} \ldots a_{i2}a_{i1}$
    - $S \Rightarrow x_{i1} Ba_{i1} \Rightarrow x_{i1} x_{i2}Aa_{i2}a_{i1} \Rightarrow \ldots \Rightarrow$
    - $x_{i1} x_{i2} \ldots x_{im} A\, a_{im} \ldots a_{i2}a_{i1} \Rightarrow x_{i1} x_{i2} \ldots x_{im}a_{im} \ldots a_{i2}a_{i1}$
  - Since (i1,i2, …,im) is a solution to PCP, $w_{i1} w_{i2} \ldots w_{im}$ will be the same as $x_{i1} x_{i2} \ldots x_{im}$, thus we have 2 separate derivations for the same string.
- G is ambiguous.

## CFG Ambiguity

- Example

| | | | | | | |
|---|---|---|---|---|---|---|
| List A | 10 | 01 | 0 | 100 | 1 | 0 |
| List B | 101 | 100 | 10 | 0 | 010 | 00 |
| | 1 | 2 | 3 | 4 | 5 | 6 |

Productions:

- $A \rightarrow 10Aa_1 \mid 01Aa_2 \mid 0Aa_3 \mid 100Aa_4 \mid 1Aa_5 \mid 0Aa_6$
- $A \rightarrow 10a_1 \mid 01a_2 \mid 0a_3 \mid 100a_4 \mid 1a_5 \mid 0a_6$
- $B \rightarrow 101Ba_1 \mid 100Ba_2 \mid 10Ba_3 \mid 0Ba_4 \mid 010Ba_5 \mid 00Ba_6$
- $B \rightarrow 101a_1 \mid 100a_2 \mid 10a_3 \mid 0a_4 \mid 010a_5 \mid 00a_6$
- $S \rightarrow A \mid B$

## CFG Ambiguity

Productions:

- $A \rightarrow 10Aa_1 \mid 01Aa_2 \mid 0Aa_3 \mid 100Aa_4 \mid 1Aa_5 \mid 0Aa_6$
- $A \rightarrow 10a_1 \mid 01a_2 \mid 0a_3 \mid 100a_4 \mid 1a_5 \mid 0a_6$
- $B \rightarrow 101Ba_1 \mid 100Ba_2 \mid 10Ba_3 \mid 0Ba_4 \mid 010Ba_5 \mid 00Ba_6$
- $B \rightarrow 101a_1 \mid 100a_2 \mid 10a_3 \mid 0a_4 \mid 010a_5 \mid 00a_6$
- $S \rightarrow A \mid B$

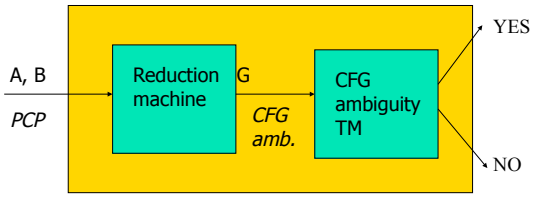
- (1,4,6) is a solution
  - Consider 101000
  - $S \Rightarrow 10Aa_1 \Rightarrow 10100Aa_4a_1 \Rightarrow 101000a_6a_4a_1$
  - $S \Rightarrow 101Ba_1 \Rightarrow 1010Ba_4a_1 \Rightarrow 101000a_6a_4a_1$

## CFG Ambiguity

- Must show that G is ambiguous iff PCP instance has a solution.
  - Assume G is ambiguous
    - A given string could have only 1 derivation starting from A and 1 starting from B
    - If there are 2 derivations, one must derive from A and the other from B
    - The string with 2 derivations will have the tail:
      - $a_{i1}a_{i2} \ldots a_{im}$ for some $m \geq 1$
      - On the A derivation the head will be $w_{i1}w_{i2}\ldots w_{im}$
      - On the B derivation the head will be $x_{i1}x_{i2}\ldots x_{im}$
      - $w_{i1}w_{i2}\ldots w_{im} = x_{i1}x_{i2}\ldots x_{im}$
      - (i1, i2, …im) is a solution to the PCP

## CFG Ambiguity

- Reduction
  - Assume CFG Ambiguity is solvable




Then PCP must be solvable…CONTRADICTION!

## CFG Ambiguity

- Finally,
  - Since PCP is unsolvable, so too is the problem of ambiguity.
  - SA ≤ HALTING ≤ MPCP ≤ PCP ≤ ambiguity

## Summary

- Solvable vs Unsolvable problems
- An unsolvable problem
  - Self-accepting
- Reducing one language to another
  - Rice's Theorem
  - Post Correspondence Problem
  - Ambiguity of CFGs.
- Questions?