Unsolvable Problems

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:
    - <u>If an algorithm or procedure exists for a given problem</u>
    - What that algorithm or procedure looks like
    - How long would it take to run this algorithm or procedure.

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 <u>algorithm</u> with <u>running on a TM</u>
  - Turing Machine is still a valid computational model for most modern computers.

Undecidability

- Informally, a problem is called <u>unsolvable</u> or <u>undecidable</u> 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 $x \in$ pal?)
    - Is a given context free language empty?

Decision Problems

- For regular languages
  1. Is the language empty?
  2. Is the language finite?
  3. Is a given string in the language?
  4. Given 2 languages, are there strings that are in both?
  5. Is the language a subset of another regular language?
  6. Is the language the same as another regular language?

Decision Problems

- For Context Free Languages
  1. Is a given string in the language?
  2. Is the language empty?
  3. Is the language finite?

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?

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 $\in$ pal?)
      - x is an instance of the probkem
    - 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 <u>decidable</u> or <u>solvable</u>. Otherwise the problem is called <u>undecidable</u> or <u>unsolvable</u>.

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 <u>are</u> accepted by the TM with w as it's encoding.
  - All encodings for TMs that <u>do</u> 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 Σ_1 and Σ_2
  - We say L_1 is reducible to L_2 ($L_1 \leq L_2$) 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

- Key facts:
  - If $L_1 \leq L_2$ 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 $L_1 \leq L_2$ 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 $w \in T(M)$?
  - 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 T_u is not recursive.
  - Instead, let's use reduction

The halting problem

- If $L_1 \leq L_2$ then
  - If L_2 is recursive then L_1 is also recursive
  - If L_1 is not recursive then L_2 is not recursive.
- Let's show that Self-Accepting can be reduced to the halting problem.

The halting problem

- Let's show that Self-Accepting can be reduced to the halting problem.
  - For an encoding of an instance of SA, I, we can define a function
    - f(I) = I'
    - Such that I' is an encoding of an instance of Halt and
    - I will be self-accepting iff I' halts.

The halting problem

- Instance of SA = (T) an encoded TM, T
- Instance of halt = (T', w) an encoded TM T' and encoded string w to run on the TM
- We want T to accept e(T) iff T' accepts w
- Let T'=T and w = e(T)
  - f(x) = (x, e(x))

The halting problem

- f(x) = (x, e(x))
  - If x is an encoding for a TM that self-accepts,
  - Then x will certainly accept e(x) as specified by f.
  - If x is not an encoding for a TM that self-accepts then either:
    - x is a bogus encoding or
    - x is an encoding for a TM that will not accept it's own encoding.
    - In either case (x, e(x)) will not be in halt.

The halting problem

- We showed that $L_1 \leq L_2$ where
  - L_1 is the set of encodings for "yes instances" of the self-accepting problem
  - L_2 is the set of encodings for "yes instances" of the halting problem.
  - We know that the self-accepting problem is unsolvable, thus, the halting problem is unsolvable.

The halting problem

- Practical considerations:
  - Since the halting problem is unsolvable, there is no algorithm to determine:
    - Given a computer program
    - Will this program always finish?
      - Alternately will it ever enter an infinite loop.

Reducing one language to another

- Important observations
  - Reduction operates on strings from languages
    - Reducing encodings of different problems
  - If $L_1 \leq L_2$ then
    - If L_2 is recursive then L_1 is also recursive
    - If L_1 is not recursive then L_2 is not recursive.
  - Questions?

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.

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

- The Self-Accepting problem can be reduced to each one of these decision problems.
- 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:

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).

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

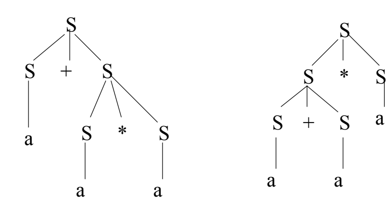
List 1	10	01	0	100	1	0
List 2	101	100	10	0	010	00
	1	2	3	4	5	6

- 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 <u>ambiguous</u> 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 Σ 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

- 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

- 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?

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
    - <u>How long would it take to run this algorithm or procedure.</u>

Next Time

- Computational Complexity and Intractability.