Undecidability/Unrecognizability

Undecidable languages: H_{TM}, A_{TM}
Unrecognizable languages: Complements of undecidable languages, AH, $\{\langle R, S\rangle \,|R \text{ and } S \text{ are TMs and } L(R) \subseteq L(S)\}$,

---

Define AH as the set of machine representations $\langle M\rangle$ such that for every input x, M halts of x.
Proof that AH is unrecognizable: Suppose it were recognizable. We will find a recognizer for the complement of the halting problem. Given inputs $\langle M\rangle$ and y, define machien M_y as follows:

1. Input x
2. Simulate M on input y for $t = \text{int}(x)$ steps
3. If M halths withing t steps: infinite loop
4. Else: halt

No run M_{AH} on input $\langle M_y\rangle$ and accept if M_{AH} accepts, which happens only when M_y always halts, which happens only when M loops on y.

Mapping Reduction

We say that "L_1 has a mapping reduction to L_2", denoted $L_1 \leq_m L_2$, if there s a computable function $f : \Sigma_1^* \to \Sigma_2^*$ such that for all $x \in \Sigma_1^*$, $x \in L_1 \Leftrightarrow f(x) \in L_2$.

Rice's Theorem

Let C be a set of languages. Consider the language L_C defined as $L_C = \{\langle M\rangle \,|L(M) \in C\}$. Then either L_C is empty, or it contains the descriptions of all Turing machines, or it is undecidable.
Proof: Suppose not for some class C. Suppose that $\emptyset \notin C$, otherwise apply the argument below to $L_{\overline{C}}$. Let M_{in} be a machine such that $\langle M_{in}\rangle$ is in L_C. Given an input $(\langle M\rangle, w)$ for A_{TM}, we construct a new Turing machine M_w that does the following: on input x, M_w first simulates the behavior of M on input w and

- If M on input w loops, then so does M_w
- If M on input w rejects, then so does M_w
- If M on input w accepts, then M_w continues with a simulation of M_{in} on input x.

In summary:

- If M accepts w, then M_w behaves like M_{in}, and M_w accepts an input x if and only if M_{in} does. In other words, $L(M_w) = L(M_{in}) \in C$ and so $\langle M_w\rangle \in C$.
- If M does not accept w, then M_w does not accept any input, and $L(M_w) = \emptyset \notin C$, which implies $\langle M_w\rangle \notin L_C$.

We have proved that $(\langle M\rangle, w) \in A$ if and only if $\langle M_w\rangle \in L_C$, and so A_{TM} would be decidable if L_C were decidable.

Godel's Incompleteness Theorems

Formalization of mathematics: a set of mathematical statements S and proofs P, and a definition of when P is a proof of S. We will make the following assumptions:

1. For every Turing machine M and string x, there is a statement $S_{M,x}$, which is computable given $\langle M\rangle$ and x, which is meant to encode the fact that M halts on input x.
2. If M halts on input x, then there is a proof P of $S_{M,x}$, which is computable given $\langle M\rangle$ and x.
3. If M on input x reaches a loop (meaning that the same configuration is encountered twice), then there is a proof of $\neg S_{M,x}$, which is computable given $\langle M\rangle$ and x.

Consistency: a formalization of mathematics is inconsistent if there is a statement S such that both S and $\neg S$ are provable.
Completeness: a formalization of mathematics is incomplete if there is a statement S such that neither S nor $\neg S$ are provable.

Theorem 1 (Godel) Every consistent formalization of mathematics that satisfies the assumptions (1), (2) and (3) is incomplete.
Proof: Suppose toward a contradiction that there is a consistent and complete system that satisfies the assumptions. Then we can define an algorithm for the halting problem. We will apply ideas from the proof that the halting problem is undecidable to construct an algorithm M_G that, when given in input its own code, provably halts if and only if it provably loops. This means that either the statements $S_{M_G,\langle M_G\rangle}$ and $\neg S_{M_G,\langle M_G\rangle}$ are both provable, in which case the system is inconsistent, or neither of them is provable, in which case the system is incomplete.
Consider the following algorithm, and call M_G the Turing machine that implements it:

1. Input: a description $\langle M\rangle$ of a Turing machine M
2. Construct the statement $S_{M,\langle M\rangle}$
3. For every string P in lexicographic order:
4. If P is a proof of $S_{M,\langle M\rangle}$ then loop
5. If P is a proof of $\neg S_{M,\langle M\rangle}$ then halt

PUT ON HOLD BECAUSE INCOMPLETENESS WAS NOT IDENTIFIED AS A MAIN MIDTERM AREA

Kolmogorov Complexity

"The first number which can be described in no fewer than fourteen words"

The pair $(\langle M\rangle, y)$ where M is a Turing machine and y is a bit string **represents** the bot string x if M on input y outputs x. $K(x)$ is defined as the smallest k for which there exists a representation $(\langle M\rangle, y)$ of x such that $|(\langle M\rangle, y)| \leq k$

Fact 1: For every n there is a string $x \in \{0,1\}^n$ such that $K(x) \geq n$.
Fact 2: For every n and every c, the probability that a random n-bit string x has Kolmogorov complexity at least $n - c$ is more than $1 - \frac{1}{2^c}$.

Theorem 3: Let $R := \{x : K(x) \geq |x|\}$ be the set of incompressible strings. R is not decidable.
Proof: Suppose M is a Turing machine that decides R. Then we construct a Turing machine M' that on input the number n outputs the lexicographically first string in $\{0,1\}^n$ which belongs to R. Let $\langle m\rangle$ denote the number n written in binary, using $\lceil \log_2 m\rceil$ bits, and consider the strings of the form $s_n := (\langle M'\rangle, \langle n\rangle)$. On the one hand, s_n, being the output of M' on input n, must be an n-bit string in R, and so $K(s_n) \geq n$. On the other hand, $(\langle M'\rangle, \langle n\rangle)$ is a representation of s_n of length $\log n + c$ for some constant c, so we must have $n \leq \log n + c$, which is false for sufficiently large values of n.

Alternate assumption (1): In our formalization of mathematics, for every binary string x and integer k, we can construct a statement $S_{x,k}$ equivalent to "$K(x) \geq k$".

Lemma 4: For every formalization of mathematics satisfying the assumptions, there is a threshold value t such that all statements of the form $S_{x,k}$ with $k > t$ are unprovable.
Proof: Consider the following algorithm:

1. Input k
2. $m := 1$
3. while (true):
4. For all strings x of length at most m:
5. For all strings P of length at most m:
6. If P is a valid proof of $S_{x,k}$, output x and halt
7. $m := m + 1$

Let M be the TM that implements the above algorithm. If k is such that there is a string for which $S_{x,k}$ is provable, then M will find such a string and output it. In such a case we have $k \leq M(k) \leq \log k + c$ for some constant c, because the output of M, by construction, has Kolmogorov complexity at least k, but the pair $(\langle M\rangle, \langle k\rangle)$ is a representation of it of length $\log k + O(1)$.

Circuits and Satisfiability

Lemma 1: Let $f : \{0,1\}^n \to \{0,1\}$ be an arbitrary Boolean function. Then there is a circuit of size $O(2^n)$ that computes f.
Proof idea: Recurrence is $f(x_1, \ldots, x_n) = (x_n \wedge f(x_1, \ldots, x_{n-1}, 1)) \vee (\neg x_n \wedge f(x_1, \ldots, x_{n-1}, 0))$ since those are two functions on $n-1$ variables. $C(n) = 2C(n-1) + O(1)$.

Theorem 2: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$ be a TM that, on inputs of length n, runs in time at most t. Then there is a circuit of size $O((|\Gamma| \cdot |Q|)^3 \cdot t^2)$ that, given an input of length n, outputs 1 if and only if M accepts x.
Proof idea: A configuration is represented using t blocks of $B = 1 + \log\Gamma + \log Q$ bits that say for each cell: what is on the tape at that cell, is the head on that cell, and if so, what state is the machine in.
For each step, every bit of output depends on at most $3B$ bits of input, so each output bit of the circuit can be computed using $O(2^{3B})$ gates, so that the entire circuit has size $O(t \cdot 2^{3B})$. This is because in one step, only the content of adjacent cells can have any effect on a cell.

Theorem 4: Circuit-SAT is NP-complete.
Proof: Let $A \in$ NP. Then A has a polynomial-time verifier V whose input has the form $\langle x, c\rangle$, where c may be the certificate showing that $x \in A$. To construct the reduction, we obtain the circuit simulating V. We fill in the inputs to the circuit that correspond to x with the symbols of w, the proposed member of A (equivalently, the proposed instance of the problem in NP). The remaining inputs correspond to the certificate c. If this circuit is satisfiable, a certificate exists, so $w \in A$. If $w \in A$, a certificate exists, so C is satisfiable. The running time of the verifier is n^k for some k, so the size of the circuit is $O(n^{2k})$.

Theorem 6: Circuit-SAT $\leq_m^p$ 3SAT, so 3SAT is NP-complete.
Proof idea: Change each gate in the circuit to a 3cnf formula, e.g. for AND: $((\overline{w_i} \wedge \overline{w_j}) \to \overline{w_k}) \wedge \cdots$, which can be converted to 3cnf.

Graph Problems

Independent Set (IS): Given a graph G and an integer k, is there an independent set of size at least k? An independent set is a subset $S \subseteq V$ such that no pair $u, v \in S$ is connected by an edge in E.
Reduction idea: 3SAT $\leq_m^p$ IS. For each clause, make a triangle. Add edges between every pair of vertices corresponding to complementary literals.

Clique: Given a graph G and an integer k, is there a clique of size at least k? A clique is a subset $K \subseteq V$ such that all the pairs $u, v \in K$ are connected by an edge in E.
Reduction idea: IS $\leq_m^p$ Clique. A subset of vertices is an independent set if and only if it is a clique in the complement graph.

Vertex Cover (VC): Given a graph G and an integer k, is there a vertex cover of size at most k? A vertex cover is a subset $C \subseteq V$ such that for every edge $(u, v) \in E$ at least one of u or v is an element of C.
Reduction idea: IS $\leq_m^p$ VC. A set S is an independent set if and only if $V - S$ is a vertex cover.

Problems About Subsets of Integers

Subset Sum: In Subset Sum the input is a sequence of non-negative integers $a_1, \ldots, a_n$ and a target value t. The question is whether there is a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i = t$.
Reduction idea: VC $\leq_m^p$ Subset Sum. Make a set of a_u for each vertex and $b_{u,v}$ for each edge. In base 4, each digit corresponds to an edge (numbered 0 through $|E| - 1$), each edge has a 1 in its corresponding digit, each vertex has a 1 for each digit corresponding to its incident edges, and everything else is zeros except for an extra leading digit (the $|E|$th digit), where all vertices have a 1. Then we want to find a sum equal to $k \cdot 4^{|E|} + \sum_{j=0}^{|E|-1} 2 \cdot 4^j$.

Partition: We are given integers $a_1, \ldots, a_n$ and the question is whether there is a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i = \sum_{j \notin S} a_j$.
Reduction idea: Subset Sum $\leq_m^p$ Partition. Consider an instance $I = (a_1, \ldots, a_n, t)$ of Subset Sum and define $A = \sum_i a_i$. Construct the instance $I' = (a_1, \ldots, a_n, a_{n+1}, a_{n+2})$ of Partition where $a_{n+1} = 2A - t$ and $a_{n+2} = A + t$. The total sum of the integers in I' is $4A$, so I' is a YES instance of Partition if and only if there is a subset that sums to $2A$.

Knapsack: In the Knapsack problem we are given integer costs $c_1, \ldots, c_n$ and volumes $v_1, \ldots, v_n$, a cost target t and a volume bound B. The question is whether there is a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} c_i \geq t$ and $\sum_i v_i \leq B$.
Reduction idea: Start with an instance $a_1, \ldots, a_n, t$ of Subset Sum and define an instance of Knapsack with n items by setting $c_i = v_i = a_i$ and $t = B = k$.

Bin Packing: In the Bin Packing problem we are given volumes $v_1, \ldots, v_n$, a volume bound B, and a target k. The question is whether we can partition the integers $v_1, \ldots, v_n$ into k subsets such that the integers in each subset sum to at most B.
Reduction idea: Partition $\leq_m^p$ Bin Packing. Given an instance $a_1, \ldots, a_n$ of partition, create an instance of Bin Packing where $v_i = a_i$, $B = (\sum_{i=1}^n a_i)/2$, and $k = 2$.

Selected Problems from Practice Midterm

Problem: Define $L = \{\langle M_1, M_2 \rangle : L(M_1) = \overline{L(M_2)}\}$. Show that L is unrecognizable.
Solution: H recognizer:

1. Input $\langle M \rangle, x$
2. Construct M_1:
3. Input z
4. simulate $M(x)$
5. accept
6. Construct M_2:
7. Input z
8. accept
9. Return $R_L(\langle M_1 \rangle, \langle M_2 \rangle)$

Prove by enumerating cases and comparing resulting languages.

Problem: Two binary strings are ϵ-close if $|x| = |y|$, and x and y differ in at most $\epsilon \cdot |x|$ positions. Prove that $Q(x) = \min_y\{K(y)|x, y \text{ are .0001-close}\}$ is not computable. You can use the fact that $\binom{n}{k} \leq \left(\frac{e \cdot n}{k}\right)^k$.
Solution: There are $\binom{|x|}{.0001 \cdot |x|}$ strings to which x is .0001-close, so x can be succinctly represented as an encoding of where it differs from some .0001-close string y using $\log \binom{|x|}{.0001 \cdot |x|} \leq \log \left(\frac{e}{.0001}\right)^{.0001|x|} \leq \frac{|x|}{100}$ bits. Since there exists an encoding for x that uses at most $|x|/100 + Q(x) + O(1)$ bits, for sufficiently large $|x|$, we have that $K(x) \leq |x|/100 + Q(x) \Rightarrow Q(x) \geq K(x) - |x|/100$. Since there exist incompressible strings, there exists a string of every length above some threshold such that $Q(x) \geq |x| - |x|/100$. Define TM M which takes in n and, if n is above the necessary threshold, returns the first x such that $Q(x) \geq |x| - |x|/100$. Then $K(M(x)) \leq \log |x| + O(1)$ since $(\langle M \rangle, x)$ represents $M(x)$. By construction, $Q(M(x)) \geq |x| - |x|/100$. By the definition of Q, $Q(M(x)) \leq K(M(x))$. So we have that $|x| - |x|/100 \leq \log |x| + O(1)$, which is false for sufficiently large $|x|$.

Selected Homework Problems