Final Exam Solutions

We fix the alphabet $\Sigma = \{0, 1\}$. A string is a member of $\Sigma^*$. A language is a subset of $\Sigma^*$. If $w$ is a string then $|w|$ denotes its length, and if $S$ is a (finite) set then $|S|$ denotes the number of elements in it. If $t$ is an integer then $0^t$ is the string of $t$ zeros and $1^t$ is the string of $t$ ones. If $x, y$ are strings then $x \parallel y$ denotes their concatenation.

Problem 1 [12 points] Briefly explain what are the following theses and how we gain confidence in them. Mention some results and discuss any general paradigms by which such results are established, but do not give details of any constructions. Your answer to each part should not be more than a page in length.

1. [6 points] The Church-Turing thesis

The Church-Turing thesis says that the class of decidable languages is unchanged under any “natural” model of computation and is that given by Turing Machines. It is a thesis because it isn’t exactly clear what “natural” means. It is a thesis, not a theorem, because we have no proof of it. (Indeed, we don’t even have a formal claim we can try to prove.)

We gather confidence in this thesis via example. We consider any specific model of computation that appears to us reasonable. We then show that the class of decidable languages is the same in this model and the Turing machine model. Models for which this has been done include: Multi-Tape Turing machines, Random Access machines, Markov Algorithms, common computer programming languages such as C, and even quantum computers.

The fact that the class of decidable languages is the same across the particular models mentioned above is not the thesis, or even a thesis at all; it is a theorem that we can state formally and that we have proved. (Or, at least, could prove with enough work.)

The paradigm used is an important one: it is called simulation. We take one model of computation and show that we can write, in another model, a simulator, or interpreter, which will be able to execute programs of the first model by substituting each of their instructions with a subroutine in the second model.

2. [6 points] The Extended Church-Turing thesis

The Extended Church-Turing thesis is the analogue of the Church-Turing thesis for polynomial time. It says that the class $\mathbf{P}$ is unchanged under any “natural” model of computation and is that given by Turing Machines. We gain confidence in it by noting that the simulations used to establish the Church-Turing thesis have polynomial overhead on most models, in particular
Multi-Tape Turing machines, Random Access machines, Markov Algorithms and common computer programming languages such as C.

The thesis is (apparently) violated by quantum computers which are able to solve, in polynomial time, problems not known to be in P for Turing Machines. But whether a quantum computer is a “natural” or even feasible device is debatable.

---

**Problem 2 [20 points]** A randomized TM $M$, given an input $x$ and having some string $R$ placed on its random tape, can not only accept or reject but output some string to its output tape. We denote this output by $M(x; R)$. If $B$ is a language and $x$ is a string we define the probability that the output of $M$ on input $x$ is in the language $B$ as

$$\text{OutPr}_M(x, B) = \frac{\left| \{ R \in \{0,1\}^{r(|x|)} : M(x; R) \in B \} \right|}{2^{r(|x|)}},$$

where $r(n)$ is the length of the random tape of $M$. We say that $A$ is randomly reducible to $B$, written $A \leq_r B$, if there exists a polynomial-time randomized TM $M$ such that for all $x \in \Sigma^*$

1. If $x \in A$ then $\text{OutPr}_M(x, B) \geq 2/3$
2. If $x \notin A$ then $\text{OutPr}_M(x, B) \leq 1/3$.

If $C_1, C_2$ are complexity classes, we say that $C_1$ randomly reduces to $C_2$, written $C_1 \leq_r C_2$, if for every $A \in C_1$ there is a $B \in C_2$ such that $A \leq_r B$. Prove the following:

1. [12 points] If $A \leq_r B$ and $B \in \text{BPP}$ then $A \in \text{BPP}$.

We have seen two analogues of this before: If $A \leq_m B$ and $B$ is decidable then so is $A$, and if $A \leq_p B$ and $B \in \text{P}$ then $A \in \text{P}$. The proof is the same, just taking error probabilities into account.

Since $A \leq_r B$ there is a polynomial-time, randomized TM $M$ such that for all $x \in \Sigma^*$

1. If $x \in A$ then $\text{OutPr}_M(x, B) \geq 2/3$
2. If $x \notin A$ then $\text{OutPr}_M(x, B) \leq 1/3$.

Let $k \in \mathbb{N}$ be a positive integer whose value we will specify later. Since $B \in \text{BPP}$ there is a polynomial-time, randomized TM $M_B$ such that for all $y \in \Sigma^*$

1. If $y \in B$ then $\text{AccPr}_{M_B}(y) \geq 1 - 2^{-k}$
2. If $y \notin B$ then $\text{AccPr}_{M_B}(y) \leq 2^{-k}$.

Let $r, r_B$ be the lengths of the random tapes of $M, M_B$, respectively, and consider the following randomized TM:

$$M_A(x; R)$$

Let $R_f$ be the first $r(|x|)$ bits of $R$

$y \leftarrow M(x; R_f)$

Let $R_B$ be the next $r_B(|y|)$ bits of $R$

If $M_B(y; R_B)$ accepts then accept else reject
We now claim that if we set $k = 3$ then for all $x \in \Sigma^*$ we have

1. If $x \in A$ then $\text{AccPr}_{MA}(x) \geq 1 - \epsilon$
2. If $x \notin A$ then $\text{AccPr}_{MA}(x) \leq \epsilon$

for $\epsilon = 11/24 < 1/2$ which shows $A \in \text{BPP}$. To see this first suppose $x \in A$. Then

$$\text{AccPr}_{MA}(x) \geq \Pr[y \in B] \cdot \Pr[M_B \text{ accepts } | y \in B] \geq \frac{2}{3} \cdot \frac{7}{8} = \frac{14}{24} > \frac{13}{24} = 1 - \epsilon.$$ 

On the other hand if $x \notin A$, then

$\text{AccPr}_{MA}(x) = \Pr[y \in B] \cdot \Pr[M_B \text{ accepts } | y \notin B] + \Pr[y \notin B] \cdot \Pr[M_B \text{ accepts } | y \notin B] \leq \frac{1}{3} + \frac{1}{8} = \frac{11}{24} \leq \epsilon$.

3. [8 points] $\text{BPP} \leq_r \text{P}$.

Let $A \in \text{BPP}$ and let $M_A$ be a polynomial-time, randomized TM recognizing $A$ with two-sided error $1/3$. We set $B = \{1\}$, which is certainly in $\text{P}$, and consider the following polynomial-time, randomized TM $M$:

$$M(x; R)
\text{If } M_A(x; R) \text{ accepts then return 1 else return 0}$$

Now for all $x \in \Sigma^*$ we have

1. If $x \in A$ then $\text{OutPr}_{M}(x, B) \geq 2/3$
2. If $x \notin A$ then $\text{OutPr}_{M}(x, B) \leq 1/3$.

**Problem 3 [20 points]** A bipartite graph $G = (L, R, E)$ has two disjoint sets of vertices: $L$, the set of “left-hand-side” vertices, and $R$, the set of “right-hand-side” vertices. Edges of the graph must cross from one side to the other, meaning each $e \in E$ has one endpoint in $L$ and the other in $R$. A pair $u, v$ of distinct vertices in $L$ are said to be siblings if there exists a vertex $w \in R$ such that $\{u, w\} \in E$ and $\{v, w\} \in E$. A subset $F \subseteq L$ of the left-hand-side vertices is said to be a family if for all distinct $u, v \in F$ it is the case that $u, v$ are siblings. Prove that the following FAMILY problem is $\text{NP}$-complete:

**Input:** Bipartite graph $G = (L, R, E)$ and integer $f \geq 0$

**Question:** Is there a set $F \subseteq L$ such that $|F| \geq f$ and $F$ is a family?

See the first page of this exam for a list of problems whose $\text{NP}$-completeness you may use without proof.

For future reference we name the language corresponding to the above problem

$$\text{FAMILY} = \{(G, f) : G = (L, R, E) \text{ is a bipartite graph and there exists a set } F \subseteq L \text{ such that } |F| \geq f \text{ and } F \text{ is a family}\}.$$
We have to show that FAMILY is \( \text{NP} \)-complete.

To show that FAMILY is in \( \text{NP} \) we need to specify a polynomial-time verifier \( V \) for it. It works as follows:

Verifier \( V((G, f), F) \)
- Parse \( G \) as a bipartite graph \( (L, R, E) \)
- Check that the following are true:
  - \( F \subseteq L \) and \( |F| \geq f \)
  - For each distinct \( u, v \in F \) there exists a \( w \in R \) such that \( \{u, w\} \in E \) and \( \{v, w\} \in E \)
- If any check fails then reject else accept

The \( \text{NP} \)-hardness of FAMILY can be proved by reduction from CLIQUE. The reduction function \( f \) takes input \((G, K)\) where \( G = (V, E) \) is a graph and \( K \geq 0 \) is an integer. It outputs the bipartite graph \( G' = (L, R, E') \) and the integer \( f \) defined as follows:
- \( L = V \) is the vertex set of \( G \)
- \( R = E \) is the edge set of \( G \)
- \( E' \) is the set of all \( \{v, e\} \) such that \( v \in e \) and \( v \in V \) and \( e \in E \), meaning each vertex \( v \in L \) is connected to the edges \( e \in R \) on which \( v \) is incident in the graph \( G \).
- \( f = K \)

Let us now check that this works. First assume \( G \) has a clique of size at least \( K \), and let \( C \) denote such a clique. Set \( F = C \). Then \( F \) is a family of size at least \( f \). Conversely, if \( F \) is a family of size at least \( f \) then it is also a clique in \( G \) of size at least \( K \).

**Problem 4 [20 points]** We say that TM \( V \) is an \( \text{NP} \)-verifier if it runs in time polynomial in the length of its first input. To any such verifier we associate the language
\[
L_V = \{ x \in \Sigma^* \mid \exists y \text{ such that } V(x,y) \text{ accepts} \}.
\]
A TM \( M_V \) solves the search problem for \( V \) if for all inputs \( x \in \Sigma^* \) its output \( y = M_V(x) \) satisfies the following: If \( x \in L_V \) then \( V(x,y) \) accepts and if \( x \not\in L_V \) then \( y = \bot \). The search problem for \( \text{NP} \) is said to be solvable if for all \( \text{NP} \)-verifiers \( V \) there is a polynomial-time \( M_V \) that solves the search problem for \( V \). Prove the following:

1. **[5 points]** If the search problem for \( \text{NP} \) is solvable then \( \text{P} = \text{NP} \).

   Since \( \text{P} \subseteq \text{NP} \) we need only show \( \text{NP} \subseteq \text{P} \). Let \( V \) be an \( \text{NP} \)-verifier. We will show that \( L_V \in \text{P} \). The assumption that the search problem is solvable for \( \text{NP} \) means there is a polynomial-time TM \( M_V \) that solves the search problem for \( V \). Let \( M \) be the TM that on input \( x \) lets \( y = M_V(x) \), rejecting if \( y = \bot \) and accepting otherwise. Then \( M \) decides \( L_V \) in polynomial time.

2. **[15 points]** If \( \text{P} = \text{NP} \) then the search problem for \( \text{NP} \) is solvable.
Let $V$ be an NP-relation. We want to show that there is a polynomial-time TM $M_V$ that solves the search problem for $V$. We set

$$A = \{ \langle x, y_1 \rangle : \exists y_2 \in \Sigma^* \text{ such that } V(x, y_1 y_2) \text{ accepts} \} .$$

In other words, $\langle x, y_1 \rangle$ is in $A$ if $y_1$ is a prefix of solution $y$ to the search problem on instance $x$. But $A \in \text{NP}$ because $V$ is computable in time polynomial in the length of its first input. Now by assumption this means that $A \in \text{P}$ so we can fix a polynomial-time algorithm $M_A$ that decides $A$. The following machine now solves the search problem for $V$:

Algorithm $M(x)$

1. $y \leftarrow \varepsilon$
2. If $M_A(\langle x, y \rangle)$ rejects then return $\bot$ and halt
3. While ($V(x, y)$ rejects) do
   1. If $M_A(\langle x, y_0 \rangle)$ accepts then $y \leftarrow y_0$ else $y \leftarrow y_1$
4. EndWhile
5. Return $y$

Algorithm $M$ starts with $y$ being the empty string, and then extends $y$ one bit at a time, using $A$ to guide its search, until a witness is found. It will halt in time polynomial in the length of $x$.

Problem 5 [15 points] A language $L$ is in the complexity class UPP if there exists a polynomial-time, randomized TM $M$ such that for all $x \in \Sigma^*$

1. If $x \in L$ then $\text{AccPr}_{M}(x) > 1/2$
2. If $x \not\in L$ then $\text{AccPr}_{M}(x) \leq 1/2$.

Prove that $\text{NP} \subseteq \text{UPP}$.

Let $L$ be a language in $\text{NP}$. We want to show that $L \in \text{UPP}$.

Since $L \in \text{NP}$ there is a verifier $V$ which takes two inputs $x, y$, halts in $p(|x|)$ steps where $p$ is a polynomial, and satisfies:

1. If $x \in L$ then there exists a $y$ such that $V(x, y)$ accepts
2. If $x \not\in L$ then $V(x, y)$ rejects for all $y$.

We define the witness set of $x$ to be

$$W(x) = \{ y : V(x, y) = 1 \} .$$

Then the above is saying that $x \in L$ if and only if $W(x) \neq \emptyset$.

To show that $L \in \text{UPP}$ we need to exhibit a randomized, polynomial time algorithm $M$ such that the conditions of the problem statement hold. The idea is that $M$ will pick a string $y$ at random, hoping it is a valid certificate that $x \in L$, meaning a member of the witness set for $x$. If so, it accepts; if not, it takes a random decision. If $x \in L$ then there is some (very small but non-zero)
chance that the chosen \( y \) is really a certificate, while if \( x \not\in L \) then this chance is zero, and this difference yields a slightly greater probability of accepting when \( x \in L \).

Let us now specify the algorithm \( M \) in more detail. It divides its random tape \( R \) into parts, \( R = yb \), where \( |y| = p(|x|) \) and \( |b| = 1 \), meaning \( y \) can be tried as a possible certificate for \( x \), and \( b \) is a single, random bit, or coin flip, \( b \in \{0,1\} \). It works as follows:

Algorithm \( M(x; R) \)
Divide \( R \) into two parts, \( R = yb \), as indicated above
If \( V(x,y) = 1 \) then accept
Else if \( b = 1 \) then accept else reject.

The accepting probability of \( M \) on input \( x \) is

\[
\text{AccPr}_M(x) = \frac{|W(x)|}{2^{p(n)}} + \left( 1 - \frac{|W(x)|}{2^{p(n)}} \right) \cdot \frac{1}{2} \\
= \frac{1}{2} + \frac{1}{2} \cdot \frac{|W(x)|}{2^{p(n)}},
\]

(1)

Now suppose \( x \in L \). Then we know that \( |W(x)| \geq 1 \), and from Equation (1) we get

\[
\text{AccPr}_M(x) \geq \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2^{p(n)}} > \frac{1}{2}.
\]

On the other hand if \( x \not\in L \) then \( |W(x)| = 0 \) so from Equation (1) we get

\[
\text{AccPr}_M(x) = \frac{1}{2}.
\]

This completes the proof.

---

**Problem 6 [13 points]** A verifier \( V \) is said to define an interactive proof for language \( L \) if there exists a prover \( P \) and a constant \( \epsilon < 1/2 \) such that for all \( x \in \Sigma^* \) the following are true:

1. **Completeness**: If \( x \in L \) then \( \text{AccPr}^p_V(x) \geq 1 - \epsilon \)
2. **Soundness**: If \( x \not\in L \) then \( \text{AccPr}^\tilde{p}_V(x) \leq \epsilon \) for all \( \tilde{P} \).

Let \( L_1, L_2 \) be languages, and \( V \) a verifier, such that \( V \) defines an interactive proof for \( L_1 \) and also \( V \) defines an interactive proof for \( L_2 \). Prove that \( L_1 = L_2 \).

By assumption there exist provers \( P_1, P_2 \), and constants \( \epsilon_1, \epsilon_2 < 1/2 \), such that \( (P_1, V) \) is an interactive proof for \( L_1 \) with error-probability \( \epsilon_1 \) and \( (P_2, V) \) is an interactive proof for \( L_2 \) with error-probability \( \epsilon_2 \). As a consequence, the following hold:

1. \( \forall x \in L_1 : \exists P : \text{AccPr}^p_V(x) > 1/2 \)
2. \( \forall x \not\in L_1 : \forall P : \text{AccPr}^p_V(x) < 1/2 \)
3. \( \forall x \in L_2 : \exists P : \text{AccPr}^p_V(x) > 1/2 \)
4. \( \forall x \not\in L_2 : \forall P : \text{AccPr}^p_V(x) < 1/2 \)
Now we assume that \( L_1 \neq L_2 \) and show this leads to a contradiction. We consider two cases.

The first case is that \( L_1 \cap \overline{L}_2 \) is non-empty. Let \( x \in L_1 \cap \overline{L}_2 \). Then from (1) and (4) we have

\[
\exists P : \text{AccPr}_V^P(x) > 1/2 \quad \text{and} \quad \forall P : \text{AccPr}_V^P(x) < 1/2.
\]

These two statements are clearly contradictory.

The second case is that \( \overline{L}_1 \cap L_2 \) is non-empty. Let \( x \in \overline{L}_1 \cap L_2 \). Then from (2) and (3) we again have the contradictory statements

\[
\exists P : \text{AccPr}_V^P(x) > 1/2 \quad \text{and} \quad \forall P : \text{AccPr}_V^P(x) < 1/2.
\]