COSC 545: *Theory of Computation*
Professor: *Calvin Newport*

Spring 2012
Georgetown University

Midterm

February 22, 2012

- **Grading Note #1:** To understand your grade, compare it to the grade distribution below. Roughly speaking, 60-79 is A/B range, 40-59 is B/C range.
- **Grading Note #2:** In grading, I tried to be generous about giving credit for partial work. I also did not end up requiring Turing Machines descriptions when showing decidability (even though the exam instructions asked for such descriptions).
- **Grading Note #3:** Please check that the grades on each individual problem are correctly reflected on your exam's title page and that the final grade on the title page is the correct sum of the individual problem grades.

Grade Distribution

30-39:	**
40-49:	**
50-59:	*
60-69:	****
70-79:	**

GRADE

Q1 : 7.5/25

Q2 : 10/10

Q3 : 20/20

Q4 : 15/15

Q5 : 10/10

Q6 : 20/20

82.5%

COSC 545: *Theory of Computation*
Professor: *Calvin Newport*

Spring 2012
Georgetown University

Midterm

February 22, 2012

- **Grading Note #1:** To understand your grade, compare it to the grade distribution below. Roughly speaking, 60-79 is A/B range, 40-59 is B/C range.
- **Grading Note #2:** In grading, I tried to be generous about giving credit for partial work. I also did not end up requiring Turing Machines descriptions when showing decidability (even though the exam instructions asked for such descriptions).
- **Grading Note #3:** Please check that the grades on each individual problem are correctly reflected on your exam's title page and that the final grade on the title page is the correct sum of the individual problem grades.

Grade Distribution

30-39:	**
40-49:	**
50-59:	*
60-69:	****
70-79:	**

GRADE

Q1 : 7.5/25

Q2 : 10/10

Q3 : 20/20

Q4 : 15/15

Q5 : 10/10

Q6 : 20/20

82.5%

Problem 1. Material Review [25 points] (5 parts)

The following questions can each be answered with a short response. These are intended to test that you reviewed the material covered in class, and should not require much new thinking.

(a) Name five operations that regular languages are closed under.

union, complementation, intersection, concatenation, star

5/5 ✓

(b) When proving that every regular language is described by a regular expression, we introduced the GNFA computational model, which is an NFA where the edges are labeled with regular expressions. When defining the state removal procedure required by this proof, we assumed that the GNFA was in a special form. What is this special form?

All states were in pairs.

0/5 ✗

(c) Sketch a brief argument for why any regular language can be recognized by a GNFA of this special form.

Because any NFA can be converted to this GNFA by splitting states and adding ε transitions between them.

0/5 ✗

(d) The following three statements are true: (1) every regular language is context-free; (2) every context-free language is decidable; and (3) every context-free language is recognizable. Choose *any two* of these statements and argue why they are true.

(1) PROOF: All regular languages can be recognized by a DFA. All context-free languages are recognized by a PDA. All DFAs are PDAs which have a null stack alphabet (or no stack operations). ✓

2.5/5

(2) All CFGs are decidable because we can create a TM which simulates the corresponding PDA on w and accept if the PDA accepts and reject if the PDA rejects (which must happen on a finite string). ✗

(e) When learning the Recursion Theorem we first defined a TM called SELF that output $\langle SELF \rangle$ (e.g., its own description). We defined this machine as the combination of two machines A and B (e.g., we said $SELF = AB$). Describe both A and B.

A is a machine which outputs ⟨B⟩ on the tape.

[illegible]

B is a machine which outputs ⟨A⟩ just prior to ⟨B⟩ on the tape. Because this is a circular definition, however, we can remove the dependency by having B output $q(\langle B \rangle)$ since we know that this is what A must be. [illegible]

0/5

Forgot description of q as the turing machine which ignores its input and outputs w.

Problem 2. Deterministic Finite Automata [10 points]

Formally define the 5-tuple for a DFA that recognizes the following language: $\{0^k \mid k$ is a multiple of $n\}$. In your answer, assume that $\Sigma = \{0, 1\}$.

DFA = $(Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2, \cdots, q_{n-1}, q_r\}$

δ: On all "0" inputs $q_i \to q_{i+1}$ except for $q_{n-1} \to q_0$ and $q_r \to q_r$

On all "1" inputs $q_i \to q_r$

$q_0 = q_0$

$F = \{q_0\}$ ✓

In brief, this DFA works by incrementing along n states, going back to zero instead of reaching q_n. If a "1" is ever found, it goes into a permanent state of non-acceptance. Accept only if $k \equiv 0 \pmod{n}$

10/10

Problem 3. Pumping Lemma [20 points] (3 parts)

Use *only* the pumping lemma to prove that the following languages are not regular. The three parts are presented in increasing order of difficulty (and point value).

(a) $A = \{ww \mid w \in \{0,1\}^*\}$

Here we consider the string $w^p = 1^{p/2} 0^{p/2}$
This string is of length > pumping length, but it cannot be broken into xy^iz such that it can be pumped.
Consider $\underbrace{\quad}_{1}\underbrace{\quad}_{0}\underbrace{\quad}_{1}\underbrace{\quad}_{0}$ either $y = \underbrace{0^k 1^k}$ or it contains uneven amounts of w_1 and w_2. In all cases $w_1 \neq w_2$ if pumped

(b) $B = \{a^{2^n} \mid n \geq 0\}$

Consider string $w = a^{2^p}$, $|w| > p$
Now we break w into xyz the only possible pumping configuration is if $y = w$ and $x = z = \varepsilon$. This can be pumped once as $yy = a^{2^{p+1}}$, however y^3 cannot be pumped as $a^{2^k} \neq y^3$ for all integral k

(c) $C = \{w_1\#w_2\#...\#w_k \mid k \geq 0, w_i \in a^*, w_i \neq w_j \text{ for } i \neq j\}$.

Let's consider the string:

$S = w_0 \# w_1 \# w_2 \# w_3 \# \ldots \# w_k$ where $k = p$
and
$w_i = a^{p+i}$

Clearly $|S| > p$

20/20

so we should, by the pumping lemma, be able to break up S into xyz and have $xy^iz \in C$.
This fails, however, because consider:

1) If y is not selected from w_0 then condition 3 $|xy| < p$ won't hold.

2) If y is any subset of w_0 then its first pump will make $w_0 \Rightarrow a^p a^l$ for $l < p$. However, since $a^p a^l = a^{p+l} = w_l$ which violates the condition

$w_0 \neq w_i, \forall i \neq 0$

$\therefore$ C isn't regular.

Problem 4. Problem Set Review [15 points]

Answer the following problem which appeared in problem set #2:

> Let $A = \{\langle R, S\rangle \mid R$ and S are regular expressions and $L(R) \subseteq L(S)\}$. Prove A is decidable.

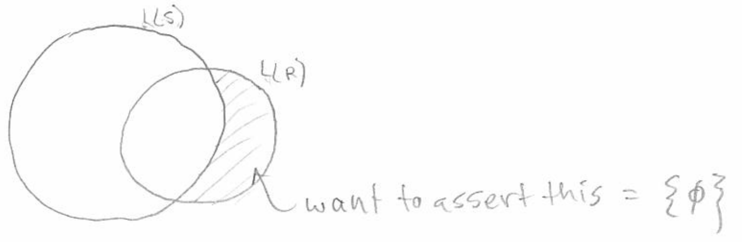
To solve this, we first note that E_{DFA} is decidable. Our goal is, then, to use E_{DFA} to decide A.
For this we first construct a language $L(C)$ which is :

$$\overline{L}(S) \cap L(R) \checkmark$$

This language is also regular because regular languages are closed under complement and intersection. We can construct a DFA for recognizing this language by converting R and S to R_{DFA} and S_{DFA} using the procedure for regular expressions → NFA and NFA → DFA and then the procedures for intersection, complement.

15/15

Finally we run E_{DFA} to check whether this DFA recognizes $\{\phi\}$ and accept if it does (and otherwise reject).



Problem 5. Undecidability [10 points]

Imagine that your boss at Microsoft is worried about algorithms that accidentally erase data, and asks you to solve the problem of determining whether a Turing Machine ever overwrites a non-blank symbol with a blank symbol during the course of its computation on any input string. Formulate this problem as a language and prove it undecidable.

As a language: $ERASE_{TM}$ $\{\langle M, w\rangle \mid$ M will overwrite a non-blank character with a blank on input w $\}$

We prove this by reduction to A_{TM}. To do this we first take M and modify it to M' such that any transition which writes a blank, writes instead a ~~with a blank overwrites it with~~ a new character '•'; This character is recognized as the same as a blank for all transitions within M'. Finally we replace

10/10

q_{accept} with two new states,* $q_{accept1}$ and $q_{accept2}$. $q_{accept1}$ always transitions by writing a blank and then going to $q_{accept2}$ at which point it halts and accepts the string. Now if $ERASE_{TM}$ existed we could feed it M' as described and output its result as the answer to A_{TM}. This is because a blank is overwritten iff the machine M accepts the input w.

* Correction, it should be 3 ~~sta~~ states in deterministic sequence:
1) write '\$'
2) Overwrite with '␣'
3) Accept.

this is a minor point.

Problem 6. Mapping Reducibility [20 points]

Prove the following for every language A:

$$A \text{ is decidable } \Leftrightarrow A \leq_m 0^*1^*.$$

(Note: $X \Leftrightarrow Y$ indicates "X *if and only if* Y"; that is, X implies Y and Y implies X.)

1) $\boxed{\Rightarrow}$ To do this we take the TM M which decides A. If it accepts $\overset{w}{\wedge}$, we set $f_m(w) = 01$. If it rejects, we set $f_m(w) = 10$. Thus the language A is ~~reducible~~ to 0^*1^*.

2) $\boxed{\Leftarrow}$ If $A \leq_m 0^*1^*$ then there exists some computable function $f_m(w)$ which takes all members of A and maps them to 0^*1^*. Since 0^*1^* is a decidable language, and $f_m(w)$ is computable, A must also be a decidable language.

20/20

COSC 545: *Theory of Computation*
Professor: *Calvin Newport*

Spring 2012
Georgetown University

Final: Solution Notes

May 10, 2012

Grade Distribution

77%

0-39:	*
40-49:	***
50-59:	*
60-69:	****
70-79:	
80-90:	*

Problem 1. Material Review [25 points] (5 parts)

The following questions can each be answered with a short response.

(a) Let $t(n)$ be a function, where $t(n) \geq n$. We learned in class that every $t(n)$ time deterministic multitape Turing machine has an equivalent $O(g(n))$ time deterministic single-tape Turing machine, and every $t(n)$ time nondeterministic Turing machine has an equivalent $O(h(n))$ time deterministic single-tape Turing machine.

Provide the smallest $g(n)$ and $h(n)$ for which we know the above to be true. For each, provide a one sentence justification for your answer.

5/5

$g(n) = n^2$ ✓ $\rightarrow$ we proved multitape can be simulated in ~~[illegible]~~ $O(t^2(n))$ time

$h(n) = 2^n$ ✓ $\rightarrow$ if it were known that it was ~~[illegible]~~ polynomial ~~tinl~~ time, P= NP. which is unknown.

(b) This question concerns the *window* used in the proof of the Cook-Levin Theorem. For each of the following four window sizes, provide a brief argument for why the theorem *would* or *would not* work if modified to use windows of this size (in the following, "$a \times b$" means a window with a rows and b columns):

1. 4×2
2. 3×3
3. $\lg n \times \lg n$
4. $\sqrt[4]{n} \times \sqrt[4]{n}$

the key for all examples is the length of ϕ_{cell} which is $O(n^{2f(m)k})$ where $f(m)$ is $O(\text{size of window})$ only if $O(\phi_{cell})$ is polynomial will the proof succeed

Doesn't store enough state $\leftarrow$ 1) Yes ~~b~~ because $O(\phi)$ would still be polynomial $\phi_{cell} = O(n^{8k})$

2) Same reasoning as (1), yes.

3) No because now $O(\phi)$ is not polynomial in length ✗ $\phi_{cell} \neq O(n^k)$ for some k

this does simplify to polynomial! ~~[illegible]~~

4) Again no, because ~~of~~ of the same reason, ϕ_{cell} now has a non-polynomial length as $n^{\sqrt[4]{n}}$ is exponential.

2/5

(c) Based on what we learned in class about the relationship between computational complexity classes, replace each __ symbol below with either $\subset$, $\subseteq$, or $=$.

$$L \subseteq NL = coNL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$$

✓ ✓ ✓ ✓ ✓ ✓

5/5

(d) In our discussion of log space, we encountered the following Theorem: If $A \leq_L B$ and $B \in L$, then $A \in L$. What is wrong with the following flawed proof argument for this theorem:

To show $A \in L$, we create a new machine C that first runs the log-space reduction implied by $\leq_L$ and then runs B on the resulting output.

5/5 What is not clear is whether the size of C is, ✓ logarithmic in size. If it contains a lot of states or tape alphabet characters, it is no longer guaranteeing that $A \in L$.

(e) Use the Space Hierarchy Theorem (and/or any of its corollaries) to prove that $NL \subset PSPACE$.

PSPACE = NPSPACE

5/5 Because $O(\lg n)$ is space computably smaller than $O(n^k)$ ✓ for all k, there must exist a language recognized by NPSPACE but not by NL, as per the Space Hierarchy Theorem.

This proves $NL \subset PSPACE$.

Problem 2. Problem Set Review[10 points] (1 part)

Let $mix(X)$, for set X of symbols, be the set consisting of every string made up of symbols from X, such that no symbol appears more than once in the string. Fix some CFG $G = (V, \Sigma, R, S)$. Prove that the language $A_G = \{X \mid X \subseteq \Sigma, \forall w \in mix(X) : w \in L(G)\}$ is in P.

The solution is to note that A_G is defined to a fixed grammar G. Since the amount of memory in a TM is infinite, all we need to do is construct a giant lookup table of whether mix(X) is in G, for all w and for all $X \subseteq \Sigma$. This is possible since mix(X) is finite for all $X \subseteq \Sigma$. Then we simply scan the string and confirm $X \subseteq \Sigma$ which takes O(n) time and that the entry for X is true on our lookup table. This algorithm runs in O(n) time and is therefore $\in P$.

- missing the explanation of conversion of $G \rightarrow$ CNF thereby limiting the size to $2|\Sigma|-1$ steps.

$5/10$

This lookup table can be constructed formally by having Q store all combinations of Σ and having δ transition to the next combination on w_i (except if $X \not\subseteq \Sigma$, in which case reject immediately) and set all correct combinations to accept at the end of the input.

Problem 3. P vs. NP [15 points] (2 parts)

Let $SPATH = \{\langle G, a, b, k\rangle \mid$ undirected graph G contains a simple path of length at most k from a to $b\}$.

(a) Theoreticians generally assume that $SPATH$ is *not* NP-complete. Provide an argument for this assumption.

This problem is solvable in polynomial time using breadth-first search. SPATH is NP-Complete iff P = NP, which theoreticians generally believe is false.

✓

(b) At the same time, theoreticians also assume that it would be very hard to formally *prove* that $SPATH$ is not NP-complete. Provide an argument for this assumption.

Again, if P = NP then all $A \in P$ are NP-Complete,* so a proof that SPATH $\notin$ NP-Complete would necessarily imply P $\neq$ NP for which a proof has remained elusive.

15/15

* as proved in the problem sets.

Problem 4. NP-Completeness [20 points] (1 parts)

Let MAXIS = $\{\langle G, k\rangle \mid$ the maximum independent set in undirected graph G is of size at least $k\}$.

Prove that MAXIS is NP-complete.

(Recall: a set of graph vertexes is *independent* if none are neighbors in the graph; the *maximum independent set* of a graph is the largest such independent set for that graph.)

(Hint: In your proof, reduce from CLIQUE, not 3SAT.)

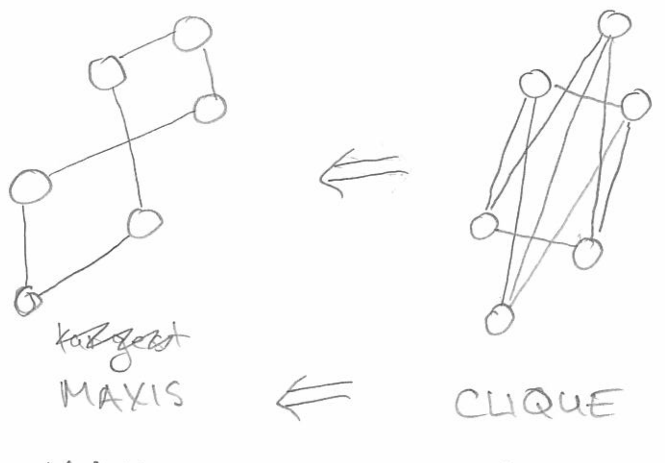
To prove this we reduce the problem from CLIQUE in the following way:

G' = for every node in the graph, let its edges be, the complement of edges in G.

$\exists$ edge in G', node$_i$ $\leftrightarrow$ node$_j$ iff $\nexists$ edge in G, node$_i$ $\leftrightarrow$ node$_j$

20/20

This reduction is linear in the size of G and so is a polynomial time reduction. Now, a clique is such that it consists of all nodes which weren't neighbors in G, so we simply run MAXIS on the inverted graph and obtain the answer for CLIQUE.



Next, MAXIS is obviously in NP because we can simply verify a list of vertices, which take polynomial time.

Problem 5. PSPACE [20 points] (1 part)

Let $A_{LBA} = \{\langle M, w\rangle \mid M$ is a linear bounded automaton that accepts $w\}$. Prove that A_{LBA} is PSPACE-complete.

(Recall: A linear bounded automaton is a deterministic single tape Turing Machine where the tape head cannot advance beyond the portion of the tape containing the input. Because the tape alphabet can only be a constant times larger than the input alphabet, such machines are limited to using memory linear in the input size—hence the name.)

(Hint: The PSPACE solution to TQBF that we briefly reviewed in class requires $O(n)$ space.)

First we show A_{LBA} is $\in$ PSPACE.
This is true because an LBA uses linear space, by definition, so is simulated in polynomial space. ✓ (Missing Sipser's qng^n upper bound!)

Second, we show $A_{LBA} \leq_{PSPACE}$ TQBF. Because TQBF's solution requires linear space, we can simulate any TQBF with an LBA, M_{TQBF}, on the input formula ϕ. M_{TQBF} is only a polynomial size in n, so any algorithm used to solve A_{LBA} can be run on $\langle M_{TQBF}, \phi\rangle$ to gain a PSPACE solution.

10/20

Missing is the TQBF algorithm, although problem does supply O(n) space for TQBF fact.

Problem 6. EXTRA CREDIT: Relativization [10 points] (2 parts)

In class, we proved the following two statements:

1. An oracle A exists whereby $P^A \neq NP^A$.
2. An oracle B exists whereby $P^B = NP^B$.

(a) What is the significance of statement 1 for the P vs. NP question?

The significance of this statement is that, relative to a class of languages, A, P is provably different from NP. Since diagonalization is invariant under relativization this means, in conjunction with (2) that it cannot be used as a proof for P = NP.

$5/10$

(b) What is the significance of statement 2 for the P vs. NP question?

This is obvious as we know there are complexity classes strictly larger than P or NP (i.e. $P \subsetneq TIME(f(n))$ and $NP \subsetneq TIME(f(n))$) so without (1) this doesn't say much about P v. NP.

X

↳ Shows we cannot get a proof for $P \neq NP$.