Geoffrey Parker - grp352
CS 341 Automata Theory
Homework 16
Due Friday, May 4 at 11:59 p.m.

This assignment reviews Chapter 21 and covers Chapter 27 and part of Chapter 28.

1) For each of the following languages L, state whether it is in D, SD/D or not SD. Prove your answer. Do not use Rices Theorem. If you claim that L is not in SD, first prove that its not in D (for practice), then prove that its not in SD. Assume that any input of the form $\langle M \rangle$ is a description of a Turing machine.

a) $\{\langle M \rangle : L(M)$ contains at least two strings$\}$.

Answer. SD/D □

Proof. Assume by way of contradiction that this language is decidable. Then there exists some machine $Oracle(\langle M \rangle)$ which decides it. Let R be a reduction from $\langle M, w \rangle$ to $\langle M\# \rangle$ as follows:

1. Erase the tape.
2. Write w on the tape.
3. Run M on w.
4. Accept.

Then let $C(\langle M, w \rangle) = Oracle(R(\langle M, w \rangle))$.
If $\langle M, w \rangle \in H$: $M\#$ accepts everything, so *Oracle* accepts.
If $\langle M, w \rangle \notin H$: $M\#$ accepts nothing, so *Oracle* rejects.
Therefore C decides H, and we know that no such machine exists, so we have a contradiction. Therefore this language is not decidable. □

b) $\{\langle M \rangle : M$ is the only Turing machine that accepts $L(M)\}$.

Answer. Decidable. □

Proof. Let $M\#$ be a machine that runs M on the universal turing machine. Then $L(M\#) = L(M)$. Therefore this language is the empty set, which is decidable. □

c) $\{\langle M \rangle : |L(M)|$ is a prime integer $> 0\}$.

Answer. $\neg$SD. □

Proof. Assume by way of contradiction that this language is semi-decidable. Then there exists some machine $Oracle(\langle M \rangle)$ which semi-decides it. Let R be a reduction from $\langle M, w \rangle$ to $\langle M\# \rangle$ as follows:

1. If x is `a` or `b` accept.
2. Erase the tape.
3. Write w on the tape.
4. Run M on w.
5. Accept.

Then let $C(\langle M, w \rangle) = Oracle(R(\langle M, w \rangle))$.
If $\langle M, w \rangle \in \neg H$: $M\#$ accepts exactly two strings, and two is a prime integer, so *Oracle* accepts.
If $\langle M, w \rangle \notin \neg H$: $M\#$ accepts everything, and infinity is not a prime integer, so *Oracle* rejects.
Therefore C decides $\neg$H, and we know that no such machine exists, so we have a contradiction. Therefore this language is not decidable. □

2) Let *Oops* be a function-computing TM defined as follows:

Input: $\langle id, c\rangle$, where id is a student id for student s and c is a course number.
Output: The grade to be assigned to student s in course c.

For whatever reason (malevolence or incompetence), it turns out that *Oops* has the property that, for all input pairs, it outputs the value `F`.

For this problem, well say that two TMs are equivalent iff they compute the same function (i.e., they halt on the same inputs and, when they halt, they output the same value).

Define $L = \{\langle M\rangle : M \text{ is equivalent to } Oops\}$. Is L in D, SD/D or ¬SD? Prove your answer.

Answer. ¬SD. □

Proof. Assume by way of contradiction that this language L is semi-decidable. Then there exists some machine $Oracle(\langle M\rangle)$ which semi-decides it. Let R be a reduction from $\langle M, w\rangle$ to $\langle M\#\rangle$ as follows:

1. Write x to a second tape.
2. Erase the tape.
3. Write w on the tape.
4. Run M on w for $|x|$ steps.
5. If M halted naturally, loop
6. Output `F`

Then let $C(\langle M, w\rangle) = Oracle(R(\langle M, w\rangle))$.
If $\langle M, w\rangle \in \neg H$: M never halts naturally, so $M\#$ outputs `F`, making it equivalent to *oops*, so *Oracle* accepts.
If $\langle M, w\rangle \notin \neg H$: there are some inputs x to $M\#$ on which M halts natually, so on these inputs $M\#$ does not output `F`, so *Oracle* rejects.
Therefore C decides ¬H, and we know that no such machine exists, so we have a contradiction. Therefore this language is not decidable. □

3) Let M be an arbitrary Turing machine.

a) Suppose that $timereq(M) = 3n^3(n+5)(n-4)$. Circle all of the following statements that are true:

i) $timereq(M) \in O(n)$.
ii) $timereq(M) \in O(n^6)$.
iii) $timereq(M) \in O(\frac{n^5}{50})$.
iv) * $timereq(M) \in \Theta(n^6)$.

Answer. ii and iii □

b) Suppose that $timereq(M) = 5^n \cdot 3n^3$. Circle all of the following statements that are true:

i) $timereq(M) \in O(n^5)$.
ii) $timereq(M) \in O(2^n)$.
iii) $timereq(M) \in O(n!)$.

Answer. iii □

4) * Let M be the Turing machine shown in Example 17.9. M accepts the language $\text{WcW} = \{w\mathtt{c}w \ : \ w \in \{a,b\}^*\}$.

a) Analyze *timereq*(M).

Answer. □

b) Is WcW in P? Why (or why do you think its not)?

Answer. □

5) Assume a computer that executes 10^{10} operations/second. Make the simplifying assumption that each operation of a program requires exactly one machine instruction. For each of the following programs P, defined by its time requirement, what is the largest size input on which P would be guaranteed to halt within a week?

a) *timereq*$(P) = 5243n + 649$.

Answer. 1,153,538,050,734 □

b) * *timereq*$(P) = 5n^2$.

Answer. 34,779,304 □

c) * *timereq*$(P) = 5^n$.

Answer. 22 □

6) * Let each line of the following table correspond to a problem for which two algorithms, A and B, exist. The table entries correspond to *timereq* for each of those algorithms. Determine, for each problem, the smallest value of n (the length of the input) such that algorithm B runs faster than algorithm A.

A	B
n^2	$572n + 4171$
n^2	$1000n \log_2 n$
$n!$	$450n^2$
$n!$	$3^n + 2$

Answer. □

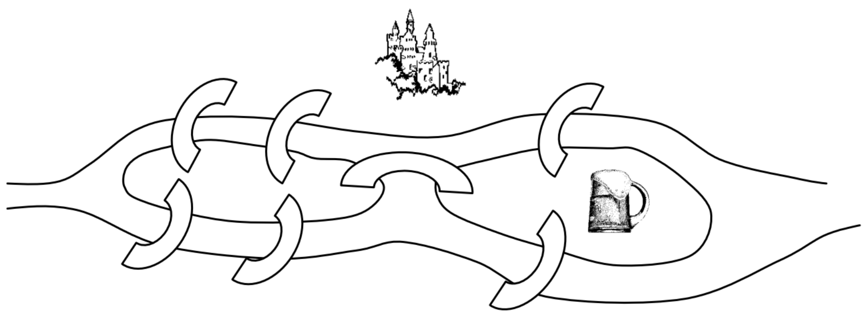
7) Show that $L = \{\langle M \rangle : M \text{ is a Turing machine and } timereq(M) \in O(n^2)\}$ is not in SD.

Proof. Assume by way of contradiction that this language L is semi-decidable. Then there exists some machine $Oracle(\langle M \rangle)$ which semi-decides it. Let R be a reduction from $\langle M, w \rangle$ to $\langle M\# \rangle$ as follows:

1. Write x to a second tape.
2. Erase the tape.
3. Write w on the tape.
4. Run M on w for $|x|$ steps.
5. If M halted naturally, loop

Then let $C(\langle M, w \rangle) = Oracle(R(\langle M, w \rangle))$.
If $\langle M, w \rangle \in \neg H$: M never halts naturally, so $M\#$ halts in $O(n)$ and therefore $O(n^2)$, so $Oracle$ accepts.
If $\langle M, w \rangle \notin \neg H$: there are some inputs x to $M\#$ on which M halts natually, so on these inputs $M\#$ never halts, so $Oracle$ rejects.
Therefore C decides ¬H, and we know that no such machine exists, so we have a contradiction. Therefore this language is not decidable. □

8) In Section 28.1.5, we described the Seven Bridges of Knigsberg. Consider the following modification:



The good prince lives in the castle. He wants to be able to return home from the pub (on one of the islands as shown above) and cross every bridge exactly once along the way. But he wants to make sure that his evil twin, who lives on the other river bank, is unable to cross every bridge exactly once on his way home from the pub. The good prince is willing to invest in building one new bridge in order to make his goal achievable. Where should he build his bridge?

Answer. From the book:

> A connected graph posseses an Eulerian path that is not a circuit iff it contains exactly two verticies of odd degree. Those two verticies will serve as the first and last verticies of the path.

Therefore he should build another bridge between the western island and the south shore, leaving only the pub island and the north shore as verticies with odd degree. □

9) * Consider the language NONEULERIAN = $\{\langle G\rangle : G$ is an undirected graph and G does not contain an Eulerian circuit$\}$.

a) Show an example of a connected graph with 8 vertices that is in NONEULERIAN.

Answer. 8 verticies, no edges. ☐

b) Prove that NONEULERIAN is in P.

Proof. All you have to do is check the degree of each node, so this is $O(n)$. ☐

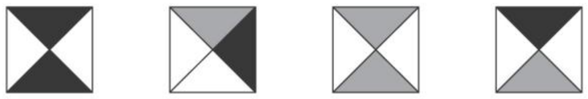
10) In Chapter 9, we showed that all of the questions that we posed about regular languages are decidable. It is shown, in Section 29.3.3, that while decidable, some straightforward questions about the regular languages appear to be hard. Some are easy however. Show that
FSM-EMPTY = $\{\langle M\rangle : M$ is a FSM and $L(M) = \emptyset\}$ is in P.

Proof. The algorith $emptyFSMgraph(M:$ FSM$)$ in section 9.1.2 decides this in polynomial time. ☐

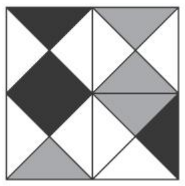
11) * Show that SUBSET-SUM = $\{\langle S, k\rangle : S$ is a multiset (i.e., duplicates are allowed) of integers, k is an integer, and there exists some subset of S whose elements sum to $k\}$ is in NP.

Proof. ☐

12) * In Section 22.3, we introduced a family of tiling problems and defined the language TILES. In that discussion, we considered the question, Given an infinite set T of tile designs and an infinite number of copies of each such design, is it possible to tile every finite surface in the plane? As we saw there, that unbounded version of the problem is undecidable. Now suppose that we are again given a set T of tile designs. But, this time, we are also given n^2 specific tiles drawn from that set. The question we now wish to answer is, Given a particular stack of n^2 tiles, is it possible to tile an $n \times n$ surface in the plane? As before, the rules are that tiles may not be rotated or flipped and the abutting regions of every pair of adjacent tiles must be the same color. So, for example, suppose that the tile set is:



Then a 2×2 grid can be tiled as:



a) Formulate this problem as a language, FINITE-TILES

Solution. ☐

b) Show that FINITE-TILES is in NP.

Proof. ☐