Simrun Heir and Michael Sanchez
I pledge my honor that I have abided by the Stevens Honor System.

Problem 1: (15 points) Prove, using the pumping lemma for context-free languages, that the language of all palindromes over the alphabet {0,1} in which the numbers of 0s and 1s are equal, is not context free. Note: We will grade this problem very closely, so make sure that your argument is complete and that no details are left implicit. The problem is not hard, the reason for this exercise is for you to write a complete and precise proof.

Answer:

Let L be the language of all palindromes over the alphabet {0,1} in which the number of 0's and 1's are equal.

Assume L is a context-free language, in which case the pumping lemma for CFLs applies

Let p be the pumping length that the pumping lemma guarantees

Consider string $s = 0^p1^{2p}0^{2p}1^{2p}0^p \in L$

Let s = uvxyz, where $|vxy| \leq p$ and $|vy| > 0$.

Since $|vxy| \leq p$ the possible cases are

Case 1: vxy contains a continuous block of 0's or 1's. When the string is pumped up, the amount of 1's are no longer equal to the amount of 0's, and in some cases it is no longer a palindrome, which exits the language.

Case 2. Since $|vxy| \leq p$, vxy can contain symbols from at most two contiguous blocks. For example, it can contain symbols from the 1st and 2nd blocks, or from the 2nd and 3rd blocks, or from the 3rd and 4th blocks. In all these cases, when we pump the string the numbers of 0's and number of 1's are no longer equal, and it is no longer a palindrome, which exits the language.

In both cases the pumping lemma is contradicted, therefore L is not a context free language.

Problem 2: (10 points) Show that the class of TM-decidable languages is closed under the following operations: union, concatenation, star, intersection, and complement.

Answer:

Union:

Yes, it is closed under union

For any two decidable languages L_1 and L_2, let M_1 and M_2 be the TMs that decide them.

Construct a TM N that decides the union of L_1 and L_2:

"On input w:

1. Run M_1 on w, If it accepts, accept
2. Run M_2 on w, If it accepts, accept. Otherwise, reject."

N accepts w if either M_1 or M_2 accepts it. If both reject, N rejects

Concatenation:

Yes, it is closed under concatenation

For any two decidable languages L_1 and L_2, let M_1 and M, be the TMs that decide them.

Construct a TM N that decides the concatenation of L_1 and L_2:

"On input w:

1. Using non-determinism, guess a partition of w that separates w such that w=xy
2. Run M_1 on x. If it rejects, reject. Otherwise move to the y portion of w.
3. Run M_2 on y. If it accepts, accept. Otherwise, reject."

N accepts w if w is of the form M_1M_2, if w does not have this form N rejects

Star:
Yes, it is closed under star
For any decidable language L, let M be the TM that decides it. Construct a TM N that decides the star of L:
“On input w:

1. Split w into parts $w_1w_2...w_n$
2. For each part w_i, run M on w_i for i=1, 2, … , n
3. If M accepts each string in w_i, accept
4. If w_i has been read and every part has been read without accepting, reject”

Intersection:
Yes, it is closed under intersection
For any two decidable languages L_1 and L_2, let M_1 and M_2 be the TMs that decide them. Construct a TM N that decides the intersection of L_1 and L_2:
“On input w:

1. Simulate M_1 on w. If M_1 halts and rejects, then reject. If M_1 loops, then N will also loop, then reject. Else, input w to M_2
2. If M_2 accepts, accept. Otherwise, reject”

Complement:
Yes, it is closed under complement
For a decidable language L, let M be the TM that decides L. Construct a TM M’ that decides the language of the complement of L, L’:
“On input w:

1. Run M over L, if M accepts, M’ rejects, if M rejects, M’ accepts”

Problem 3. (10 points) Show that the class of TM-recognizable languages is closed under the following operations: union, concatenation, star, and intersection. Is it closed under complement?

Answer:
Union:
Yes, it is closed under union
For any two Turing-recognizable languages L_1 and L_2, let M_1 and M_2 be the TMs that decide the. We construct a TM N that recognizes the union of L_1 and L_2:
“On input w:

1. Run M_1 and M_2 alternately on w step by step. If either accepts, accept. If both halt and reject, reject.”

If either M_1 or M_2 accepts w, N accepts w because the accepting TM arrives to its accepting state after a finite number of steps. Note that if both M_1 and M_2 reject and either of them does so by looping then N will loop.

Concatenation:
Yes, it is closed under concatenation
For any two Turing-recognizable languages L_1 and L_2, let M_1 and M_2 be the TMs that recognize them. Construct a TM N that recognizes the concatenation of L_1 and L_2:
"On input w:

1. Using non-determinism, guess a partition of w that separates w such that w=xy
    a. Note that x represents strings from L_1 and y represents strings from L
    b. Can be rewritten as $w=L_1L_2$
2. Run M_1 on the x part of w . If it halts and rejects, reject. Otherwise move to the y part
3. Run M_2 on the y part of w. If it accepts, accept. If it halts and rejects, reject"

Star:
Yes, it is closed under star
For any Turing-recognizable language L, let M be the TM that recognizes it. Construct a TM N that recognizes the star of L:
"On input w:

1. Non-deterministically split w into parts $w_1w_2\ldots w_n$
2. Run M on w_i for all i=1, … , n. If M accepts all parts of w, accept. If N halts and rejects for any part w_i, reject."

By cutting w into strings $w_1w_2\ldots w_n$ such that each string wi is an element of L, then there is a way for N to either recognize the string and accept w, which is done in a finite number of steps

Intersection:
Yes, it is closed under intersection
For any two Turing-recognizable languages L_1 and L_2, let M_1 and M_2 be the TMs that recognize them. Construct a TM N that recognizes the intersection of L_1 and L_2:
"On input w:

1. Run M_1 on w. If M_1 halts and rejects, reject. If it accepts, proceed to the next step.
2. Run M_2 on w. If M_2 halts and rejects, reject. If it accepts, accept."

$M_1\cap_2$ accepts a string w only if both M_1 and M_2 accept, thus w belongs to $L_1 \cap L_2$

Complement
No, it is not closed under complement.

Assume that Turing-recognizable languages are closed under complement.
If language L is Turing-recognizable, so is L', the complement to L.
If both L and L' are Turing-recognizable, then L is a decidable language, as proved in problem 2.
This creates a contradiction as the language L' is a Turing-recognizable language, but is not decidable, like L.
Therefore Turing-recognizable languages are not closed under complement

Problem 4. (10 points) Show that every infinite TM-recognizable language has an infinite decidable subset.
Notes:

- Thm 3.16: Every nondeterministic Turing machine has an equivalent deterministic Turing machine
    - Corollary 3.18: A language is Turing-recognizable iff some nondeterministic Turing machine recognizes it
    - Corollary 3.19: A language is decidable iff some nondeterministic Turing machine decides it
- Thm 3.21: A language is Turing-recognizable iff some enumerator enumerates it
- Recognizable vs decidable
    - Recognizable means the language can be accepted in finite time
    - Decidable means the language can be accepted or rejected in finite time
    - Every decidable language is Turing-recognizable but not every recognizable language is Turing-decidable
- Relationship among classes of languages
    - Regular $\subset$ Context-free $\subset$ Decidable $\subset$ Turing-recognizable
- Ch 3 problem 3.18: Show that a language is decidable iff some enumerator enumerates the language in standard string order
    - Note that enumerating a string in standard string order is lexicographical order
    - Needs to be a case in the proof to account for finite and infinite languages
- Proof sketch
    - Construct an enumerator that simulates the enumerator for an infinite TM-recognizable language, which will enumerate a set of string in lexicographical order, where the set of string enumerated are a subset of the set of strings in enumerated from infinite TM-recognizable language.
        - Note the strings produced in the subset are apart of an infinite TM-recognizable language that is the subset of another infinite TM-recognizable language
    - Once this enumerator is made, which makes an infinite subset of strings, insert a proof to problem 3.18 to show that the strings produced by the language of the simulating enumerator is decidable, making the language of that enumerator, which is already an infinite TM-recognizable language, an infinite decidable subset of an infinite TM-recognizable language
        - This step is a combination of the proof for problem 3.18 and the definitions of TM-decidable and recognizable languages

Answer:
Let L be an infinite TM-recognizable language.
Let TM M recognize language L.
By theorem 3.21 there is some enumerator that enumerates it.
Let E be the enumerator that enumerates all strings in L.
Construct a new enumerator F that prints a subset of L in lexicographical order.
F = “Ignore the input

1. Simulate E
2. When E prints the first string w_1, print w_1 and let $w_{prev} = w_1$
3. Continue simulating E

4. When E is ready to print a new string w, check to see if w is longer than wprev
    a. If w is longer than w_{prev}, print w and let w_{prev} = w, the string that was just printed
    b. If w is not longer than wprev, do not print w
5. Repeat steps 3 & 4 until E is done being simulated “

F, as constructed, only prints strings in L, and is therefore a subset of L.
L is infinite, so there will always be strings in L that are longer than the current wprev.
Due to this, when E eventually prints one of these strings so will F, causing wprev to be updated, continuing the process.
Since L is an infinite language, this process will repeat infinitely, therefore the language of F produced through this process will also be infinite.
Note that the language of F produces an infinite subset of the strings L produces

In order to prove the language of F is decidable, we must show that a language is decidable iff some enumerator enumerates the language in standard string order (a.k.a. Lexicographical order)

Proof:

- If the language N is decidable by some TM M, the enumerator operates by generating the string in lexicographical order, testing if each string is a member of N using M, and if it is it prints the string
- Two cases, N is finite and N is infinite
    - N is finite
        - If N is finite it is decidable because all finite languages are decidable
    - N is infinite
        - The decider, TM M, operates similarly to F, but has distinctions as follows
            - Takes in an input, w, and M runs the enumerator E
            - E then enumerates all strings in N lexicographically
            - This continues until a string lexicographically after w appears and once that string appears one of two things can occur
                - If w has appeared in the enumeration already, then accept
                - If w has not appeared in the enumeration already, then reject
    - Note that when N is finite it is possible to determine if the language is decidable without an enumerator, therefore an enumerator is only necessary to construct a decider when N is infinite

Since the language of F is lexicographically generated, it is a decidable language as shown in the proof above.
Thus, the language of F is an infinite decidable subset of L.
Therefore, every infinite TM-recognizable language has an infinite decidable subset.