Pledge: I pledge my honor that I have abided by the Stevens Honor System. -Eric Altenburg

1: Show that the class of TM-decidable languages is closed under the following operations: union, concatenation, star, intersection, and complement.

(Union)

We can construct TM's M_1 and M_2 to decide on languages L_1 and L_2 respectively. Then we make a new TM M' that will then decide the union of L_1 and L_2.

"On input w:

1. Run M_1 on input w, and if it accepts, then M' ACCEPT.
2. Run M_2 on input w, and if it accepts, then M' ACCEPT. Otherwise, M' REJECT."

With this procedure, M' will accept the union of L_1 and L_2 if M_1 or M_2 accepts, else, if none of them accept, then M' will reject. Also, it can be said that if only one of the TM's accept, then M' will accept, therefore, either M_1 or M_2 must accept.

(Concatenation)

We can construct TM's M_1 and M_2 to decide on languages L_1 and L_2 respectively. Then a new 3-tape TM M' is made that will decide the concatenation of L_1 and L_2.

"On input w:

1. First, nondeterministically split the string as to allow multiple different variations run on M' as the string can be split in many ways. For now, call the split string $w = w_1w_2$, then copy w_1 onto the second tape of M' and w_2 onto the third tape of M'.
2. On the second tape where w_1 is, run M_1 on it. If it accepts, then proceed to stage 3. Otherwise, M' REJECT.
3. On the third tape where w_2 is, run M_2 on it. If it accepts, then ACCEPT. Otherwise, M' REJECT."

M' is a nondeterministic decider due to the fact that both M_1 and M_2 are deciders and the L(M) is the concatenation of L_1 and L_2. Since you can simulate a 3-tape TM on a single tape deterministic decider, the concatenation is a closed operation.

(Star)

Construct a TM M_1 that decides on the language L_1. Using a 2-tape TM M':

"On the input w:

1. Nondeterministically select the leftmost unread part of the input string and place it onto the the second tape.
2. Run M_1 on the second tape's current string. If it accepts, and the entirety of w has been processed, then M' ACCEPT. If not all of w has been processed and M_1 accepts, clear the second tape and go to stage 1. If M_1 rejects, M' REJECT."

M' is a nondeterministic decider due to M_1 also being a decider, and the language of M' is the start operator on L_1. Since any 2-tape TM can be simulated with a single tape TM, M' shows that it is a decider for the star operation therefore making it closed.

(Intersection)

Similar to union, we construct two TM's M_1 and M_2 to decide on languages L_1 and L_2 respectively. Then we make a new TM M' that will then decide the intersection of L_1 and L_2.

"On input w:

1. Run M_1 on input w, and if it accepts, then move onto stage 2. Otherwise, M' REJECT.
2. Run M_2 on input w, and if it accepts, then M' ACCEPT. Otherwise, M' REJECT."

For this, both TM's M_1 and M_2 must decide on the intersection of L_1 and L_2, therefore, M' will only accept it both of the TM's accept, and if only one or none accept, then M' rejects.

(Complement)

Have a TM M_1 that decides on the language L_1. Now, have another TM M' that decides on $\overline{L_1}$:

"On input w:

1. Run M_1 on the input w, if it accepts, M' REJECT. Otherwise, if it rejects, M' ACCEPT.

This proves that the complement is closed under decidable languages because the complement is everything that is not in the language, therefore, so long as the TM that decides on L_1 is accepted, M' will reject, and if it rejects, M' will accept.

2: Show that the class of TM-recognizable languages is closed under the following operations: union, concatenation, star, and intersection. Is it closed under complement?

(Union)

For two Turing-recognizable languages L_1 and L_2, let the TM's M_1 and M_2 recognize these languages respectively. We construct a TM M' that recognizes the union of L_1 and L_2.

"On input w:

1. Alternatively run M_1 M_2 on w step by step. Then, if either of M_1 or M_2 accept, M' will ACCEPT. If both of them halt and reject, then M' REJECT."

If either M_1 or M_2 accepts w, then M' will accept w because it will arrive to its accept state in a finite amount of steps. If both M_1 or M_2 rejects and either of them do so by means of looping, then M' will loop.

(Concatenation)

We construct two TM's M_1 and M_2 to recognize languages L_1 and L_2 respectively. Construct a nondeterministic TM M' that will recognize the concat. of L_1 and L_2.

"On input w:

1. Split the input w into w = w_1w_2.
2. Run M_1 on w_1, and if it halts and rejects, then M' will REJECT.
3. Run M_2 on w_2, and if this accepts, then M' will ACCEPT. Otherwise, if it halts and rejects, then M' will REJECT."

This is very similar to proving it is decidable, however, now the condition of whether or not the TM halts must be taken into consideration.

(Star)

For a Turing-recognizable language L, we can construct a nondeterministic TM M' that will recognize L*.

"On input w:

1. Nondeterministically cut w into into several substrings $w_1w_2...w_n$.
2. Then run M' on every substring w_i for every single i.
3. If M' accepts all substrings, then ACCEPT. Otherwise, if for any substring M' halts or rejects, then REJECT."

Similar to the decidable proof above, so long as w can be broken into substrings that are still in the language, then M' has a path that allows it to accept in a finite number of steps.

(Intersection)

For two Turing-recognizable languages L_1 and L_2, let the TM's M_1 and M_2 recognize these languages respectively. Create another TM M' that recognizes that intersection of L_1 and L_2.
"On input w:

1. Run M_1 on w. If it halts or rejects, then M' will REJECT. If it accepts, then proceed to stage 2.
2. Run M_2 on w. If it halts or rejects, then M' will REJECT. If it accepts, then M' will ACCEPT."

This is similar to the union proof found above, however, both the TM's M_1 and M_2 must accept, therefore, it shows that w belongs to the intersection of the two languages.

(Complement)

This can be accomplished by a proof by contradiction. Suppose there is a language L that is Turing-recognizable by TM M_1, and its complement, $\overline{L}$, is Turing-recognizable by TM M_2. We can show that the language L is actually decidable by running M_1 and M_2 in parallel. For some input w, we will run M_1 and M_2 on each input by alternating back and forth for each step, and when either one of them accepts, the outcome will be returned. Whether or not w is or is not a member of the language, the process will result in a halt.

"On input w:

1. For the case where the input w is in the language L, then M_1 accepts after k-amount of steps, and the overall process will halt after 2k steps for some value k.
2. For the case where the input w is not in the language, then it can be said that it belongs to the complement of it $\overline{L}$, and if this is the case, it will play out to a similar fashion as stage 1. M_2 will accept after k-amount of steps, and the overall process will halt after 2k steps for some value k.

This is a contradiction as this shows that the language is actually decidable, as opposed to the previous assumption of it being Turing-recognizable.

3: Show that a language is decidable if and only if there is an enumerator which prints out strings in the language in standard string order (lexicographic, in order of increasing length).

(-)

Take a decidable language L and a TM M. Also, have the lexicographic ordering of the strings in Σ^* be represented by a series of variables such as $s_1 s_2 ... s_n$. From this, the enumerator E will:

"Ignore the input

1. For all i, run M on s_i.
2. If M accepts at any point, print out the string s_i, however, if M rejects it, move on."

Any concern of this looping is immediately taken care of as the language L is deemed as decidable. Also, this is clear that the enumerator E will print out all strings in L in lexicographic order.

(-)

Once again, consider a language L that is enumerated in lexicographic order by an enumerator E. We now suppose L to be infinite as if it were finite, then it would be clear that it would be decidable. The TM M that decides L will:

"On input w

1. Wait for E to print out a string s.
2. M will ACCEPT if the string s is the same as the input w.
3. M will REJECT if the string s is greater than the input w.
4. If the string s is less than the input w, then go to stage 1."

This will halt at some point as E will never run out of strings, and there are only a finite amount of strings that are $\leq$ w.

4: Show that every infinite TM-recognizable language has an infinite decidable subset.

Have the language L be Turing-recognizable with an enumerator E that enumerates all strings in L with the possibility of repetition. We then construct another enumerator E' that will output a subset of L in lexicographic order:

"Ignore the input

1. Simulate E and whenever E prints out the first string s_1, have E' print s_1 as well and let the previous string be s_1.
2. Continue simulating E.
3. If E prints another string s_{new}, check to see if it is longer than the previous string to make sure that the lexicographic order still holds. If it is, then print the string s_{new} and set the previous string to be s_{new}.
4. Continue to stage 2."

To help prove this, we use information from problem 3. The E' mentioned here will exclusively print strings in L making its language a subset of L. Now, since L is infinite there will always be strings in the language L that are longer than the previous string allowing E and E' to print the string while also updating their previous strings. This means that E' is infinite as well and since it prints in lexicographic order, the language is decidable as proved in the previous problem. Therefore, one can say that the E' 's language is an infinite decidable subset of L thus proving the problem.

Optional Problem 1: Prove that the language $A \backslash B = \{w : wx \in A, x \in B\}$, where A is a CFL and B is regular is a CFL.

Couldn't figure this one out in time

Optional Problem 2: Show that a language can be recognized by a deterministic queue automaton iff the language is Turing-recognizable.

You must prove in two directions:

(One Direction [not like the band])

Given the queue automata, we can simulate a TM M:

Write a "\$" to the queue to indicate the left hand side of M's tape.Consider the head of M to always point to the right hand side (RHS) of the queue. Now say there is a symbol X on the right hand end of the queue, and from this two things can be done:

1. X $\rightarrow$ Y, L

   Pull X out of the queue and push Y onto the queue. This will shift all the symbols in the queue to the right one bit.

2. X $\rightarrow$ Y, R

   Write a mark on the left end of the queue in order to make that symbol unique. Then pull X and push Y into the queue. Pull out the right hand end element and push that into the queue. Repeat this until the unique symbol marked before reaches the right end of the queue, then remove the mark.

From these procedures, a TM M was simulated by using the operations of a queue automata.

(Other Direction)

Given a TM M, we can simulate a queue automata:

To simulate the queue on the input tape of M, write a "$" on the left hand end of the tape to identify the left hand end. Two queue automata operations are push and pull:

1. Push

   To Push a new element on the left hand end of the tape, move all the symbols except for the "$" over to the right by one spot and make the head of the tape point to the new empty space to the right of the "$" and place the symbol to be pushed there.

2. Pull

   Mark the elements that need to be pulled out of the queue on the input tape this way we can ignore them.

With this, a queue automata was simulated with a TM M.

Due to the proofs in both directions, we can say that a language can be recognized by a deterministic queue automaton iff the language is Turing-recognizable.