P1

We know that,

1. Every Turing recognizable language has an enumerator: a TM that prints out the strings in the language one by one.
2. If a language has a lexicographic enumerator - a TM that enumerates the strings in the language in lexicographic order, then it is decidable.

Let L be an infinite Turing recognizable language. Let M be the enumerator that enumerates all string in L (in some order, possibly with repetitions). We construct another enumerator M' that prints a subset of L in lexicographic order.

Ignore the input.

1. Simulate M. When M prints its first string w1 print w1 and let prev_w = w1.
2. Continue simulating M.
3. When M is ready to print a new string w check to see if w is longer than prev_w, this ensures w occurs after prev_w in lexicographic order. If so then print w and let prev_w = w otherwise do not print w.
4. Go to step 2.

It is clear that M' as constructed above only prints strings in L therefore its language is a subset of L. Since L is infinite there will always be strings in L longer than the current prev_w - M will eventually print one of these and so will M' and update prev_w. Therefore the language M' is also infinite. Finally since M' only prints strings in lexicographic order its language is decidable. Thus the language of M' is an infinite decidable subset of L.

————————

P2

Note that unlike a PDA a Queue automaton can cycle through its queue by pulling on the front and pushing on the back. While it does this it can look for a particular symbol x and change it by pushing a modified symbol y or a word of more than one symbol by pushing y when it pulls x. So after each step of the TM our queue contains a string which consists of the entire tape with the TM's current position marked with an additional symbol indicating its state. We also use markers for the left and right ends of the tape.

It's easy to see how by cycling through the queue the queue automaton can carry out one step of the Turing machine by reading its current state and the current tape symbol, pulling them off the front, pushing symbols corresponding to the new state and new symbol on the back and then continuing cycling through. In order to move the machine to the left in which case the TM marker should be pushed before the tape symbol we read before we can keep a one symbol buffer instead of immediately pushing the symbol we pulled.

If the queue has length L, each step of the TM then takes O(L) steps of the queue. Since after T steps of the TM L is at most n + T where n is the length of the input, the total time to simulate T steps is O(L T) = O((n + T) T) = O(T2).

The equivalence between the queue automaton Q and the Turing machine M is required to be shown. It means it is required to be shown that the language that can be recognized by the automaton it can also be recognized by the TM and vice versa. This can be done by showing simulation which means that simulate the automaton Q to behave exactly like the TM M and vice versa.

dtrived5@stevens.edu

TM can simulate a Queue automaton easily. Consider the entire tape as a queue. One by one each symbol is altered and the movement of the tape takes place to the right. If more than one symbols are to be pushed in the queue, then it is done by shifting contents to the right and inserting symbols. If the end of the tape is reached the leftmost symbol of the tape is approached.

The TM can be simulated by the Queue automaton. Insert a left end marker # first to the queue. Then push all the symbols of tape in queue and finally a blank symbol. We are pushing symbols to the left and pulling from right. Whenever the blank symbol is read and overwritten by some other symbol push a blank symbol to the queue before pushing overwritten symbol. E.g. ABCDEF_ was the configuration of queue and TM writes a G in place of blank symbol _. Then new configuration is G_ABCDEF. Whenever a symbol is read and pushed to the queue push a symbol which tells queue that next symbol to be read is actually last symbol read. E.g. In ABCDEF, F is read and overwritten by symbol G then insert last symbol read marker | before pushing G, i.e. G|ABCDE. Read symbol E, push H, and move to left then last symbol read is E so the new configuration should be H|G|ABCD. To remove marker before G consider next symbol to read is D. Insert $ as a next symbol to read marker. E.g. D$H|G|ABC. Roll queue. CD$H|G|AB. Roll until we get | on rightmost of the queue. E.g. ABCD$H|G|. Remove marker | and roll until we reach $. H | GABCD $. Remove $ e.g. H | GABCD. If TM is to move to the last symbol read then roll until first | is exposed to right and remove rightmost |. E.g. ABCDH|G.

————————

P3

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

Assume we have a PDA X for A and DFA Y for B. To build a PDA Z for A\B the states of Qz of Z are Qz x Qy.

Phase 1: Read the input word and advance in X, states of Y are ignored. This corresponds to the word w of language.

Phase 2: When the end of w is reached, perform ε-transitions in order to reach an accepting state of X x Y. Guess a word x that will continue to advance in X and start to advance in Y and that has to reach acceptance state in both automata simultaneously. If there is an accepting run of this automaton then the input word w is in A\B since it was a correct guess. If x exists then it can be guessed by automata Z.

It can be proved by rational transductions. Let X be the alphabet. Relation τ defined by τ(u)=uB is a rational transduction. Its graph $(\Sigma_{x \in X}(x, x))^*(\{1\}XB)$ is a rational subset of X* x X*. This is the place where the hypothesis that B is regular is mandatory.

$$A \setminus B = \{u \,|\, \tau(u) \cap A \neq \phi\} = \tau^{-1}(A)$$

Where τ^{-1} denotes the inverse of the relation τ which is also a rational transduction. Finally it suffices to use the fact that if L is context free and σ is a rational transduction then $\sigma(L)$ is context free.

————————

P4

L = { <M>: M is a TM whose language L(M) satisfies property P }

If P is any property that satisfies conditions (a) If L(M1) = L(M2) then <M1> and <M2> are both in L or are both not in L and (b) There are TMs in L and TMs that are not in L, then L is undecidable. In other words every interesting property of TMs that depends on the language of the TM is undecidable.

1. Let P be any property of the language of a TM that satisfies conditions (a) and (b).
2. Assume that L is decidable and let TM D decide L.
3. Since condition b is satisfied by P let $< M > \in \mathscr{L}$ and $< N > \notin \mathscr{L}$
4. Now consider the following TM X:
   1. On input w: compute own description <X>
   2. If D accepts <X> then reject
   3. If D rejects <X> then accept

Since in both cases X contradicts D we conclude that L is undecidable. To prove that L is undecidable consider an undecidable input instance <X> is created whose output is contradictory to the fact that L is decidable. Output of D on input <X> cannot be decided. If D accepts <X> then it means X satisfies the property but then in such case X reject the input w and hence X satisfy the property P is contradictory. Similarly if X does not satisfy the property P then D rejects <X> then X accepts the input and hence X does not satisfy the property P. Thus any description of Turing machine D on input <X> will be contradictory which shows that decidable TM D cannot exist for L.

————————

P5

(a) Show that $\leq_m$ is a transitive relation

Transitive relations are those in which one element is related to the second element and second element is related to the third element. In this situation the first element should be related to the third element. If the condition is fulfilled then it can be said that the relations is transitive. Let f be the function that reduces A to B, i.e. $A \leq_m B$ by f and let g be the function that reduces B to C, i.e. $B \leq_m C$ by g. Now consider a composition function that can be considered as the mapping function between f and g. $h(w) = g(f(w))$. Now find the mapping between these two functions it may be required to create a TM that computes h as follows:

1. Simulate a TM on input w and call the output $f(w)$, this will simulate a TM for g on $f(w)$
2. The output of this input can be considered a function that is $h(w) = g(f(w))$

Then $w \in A \iff f(w) \in B$. Furthermore, $x \in B \iff g(x) \in C$. It follows that for every $w \in A, g(f(w)) \in C$, and furthermore, for every $x \in C$, there exists a $y \in B$ such that $g(y) = x$, and for every $y \in B$, there exists a $w \in A$ such that $f(w) = x$, so that for every $x \in C$, there exists a $w \in A$ with $g(f(w)) = x \in C$. Thereby $h(w) = g(f(w))$ is a computable function. Here the input w on A is directly dependent on h(w). Thus $w \in A \iff g(f(w)) \in C\ s.t.\ A \leq_m C$.

dtrived5@stevens.edu

(b) Show that if A is TM-recognizable and $A \leq_m \bar{A}$ then A is decidable

We know that,

1. If $A \leq_m B$ and B is decidable then A is decidable.
2. If $A \leq_m B$ and A is undecidable then B is undecidable.
3. If $A \leq_m B$ and B is recognizable then A is recognizable.
4. If $A \leq_m B$ and A is unrecognizable then B is unrecognizable.
5. If $A \leq_m B$ then $\bar{A} \leq_m \bar{B}$.

Now we prove that a language is decidable iff both it and its complement are recognizable. A decidable language is recognizable. Moreover if a language is decidable then so is its complement and hence that complement is recognizable. Suppose a language L and its complement are recognizable. Let A be a recognizer for L and B for its complement. A decision method for L is obtained by running A and B in parallel on a given input string. In case A accepts the string it is accepted as a member of L and in case B accepts it is rejected as member of L. One of this outcomes will occur within a finite amount of time.

If $A \leq_m \bar{A}$ then we also have $\bar{A} \leq_m A$. Since A is recognizable then because of the result of $A \leq_m B$ with A being recognizable implies B recognizable then $\bar{A}$ is recognizable. But we know that if a language and its complement are both recognizable then it is decidable.