Problem 1 Solution

(possible solution)

Consider $S \subset V$ where $S \neq \emptyset$. If we now have a stream E that has only the edges necessary to minimally connects S, call this connected component K_1, and also *separately* minimally connects $V \setminus S$, which we can call K_2, then E is distinguishable from every other similarly constructed stream:

Assume E is constructed as was described from some $S \subset V$ as was E' for some other $S' \subset V$ (note that E spawns the two separate connected components K_1 and K_2, and E' yields K_1' and K_2') such that $S' \neq S$ and $S' \neq V \setminus S$. Because of this, $\exists v_1 \in S$ such that $v_1 \notin S'$ and $\exists v_2 \in V \setminus S$ such that $v_2 \notin S'$. By definition this means $v_1, v_2 \in V \setminus S'$ and so the undirected edge $\{v_1, v_2\}$ will stay within K_2' without touching K_1' whereas it connects K_1 and K_2. Therefore, $G = (V, E \cup \{v_1, v_2\})$ is a connected graph but $G' = (V, E' \cup \{v_1, v_2\})$ is not.

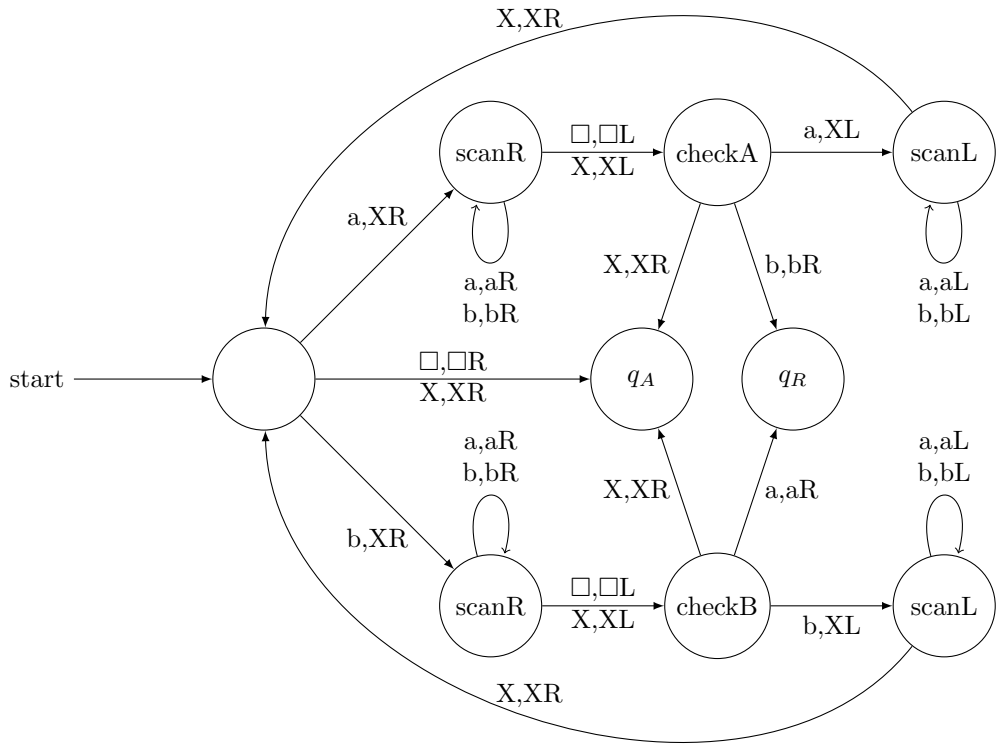
By adding $\{v_1, v_2\}$ to our stream, we can then distinguish E from E'. So $S \subset V$ is distinguishable from any other subset S' so long as $S' \neq V \setminus S$. This gives us roughly 2^n possible subsets and thus distinguishale strings (of course we have to divide by two for double counting and account for the empty set and such but that goes away asymptotically) and so we need $\Omega(n)$ bits of memory.

Note that this is only one possible solution and there are many possibilities of this flavor that can use star graphs and path graphs to get the same type of subset argument happening.

Problem 2 Solution

General idea:

Have the TM read the first letter, remember it (by transitioning to the correct state), write an X over it, scan to the right until seeing the first blank (or X) and check that the preceding character (the last part of the input) matches the first and write an X over it if it does and reject if it doesn't. Then go to the left until an X is seen to repeat the whole process above on the next letter past the X. If we see an X automatically as we try to restart the process then we've finished and should accept.



Problem 3 Solution

(possible solution)

To solve this we will show that we can only have a finite number of distinguishable strings in a read-only Turing Machine and use the Myhill-Nerode theorem to conclude that the language that is decided by that Turing Machine is regular. And since our read-only Turing Machine will be arbitrary, all such machines can only recognize regular languages.

Assume we have a read-only Turing Machine, $M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$, that decides some language. For each string x we will associate with it a function $f_x : Q \cup \{begin\} \to Q$. Intuitively, this function will tell us how our machine will "behave" on x. We want to get to the point where we can argue that if two strings define the same functions (that is M "behaves" the same on them), then they will be indistinguishable. Finally, since there are only finitely many possible functions (since Q is finite) there can only be that many distinguishable strings and we will be done.

So let's start defining f_x: it's purpose is to record all the essential information about x for the Turing Machine. We must ask how M "behaves" on x. Does M halt without ever moving past the end of x? If yes, does it accept or reject? If not, in which state is M in the first time it"crosses" beyond x? M is completely deterministic (even though it can move left and right this entire time) and the answers to all these questions are completely defined by it. We can record the answer to these questions in $f_x(begin)$, as more rigorously defined later.

While this gives us a lot of information about what M does while parsing x, that may not be the end of

the story for a read-only Turing Machine. That is, it can *re-enter* the string x after having moved past it since it can move left. If it is on state q when it transitions back into x then we want to know, again, how it will behave on x. Does M halt without ever moving past the end of x? If yes, does it accept or reject? If not, in which state is M in the when it "crosses" beyond x again? Since we re-entered on state q we can record on which state it deterministically re-exits or halts on x with $f_x(q)$. And so, formally

- $f_x(begin)$ is the state of machine M when, started on an input of the form xz, reaches the end of x for the first time and then makes a further move to the right. More formally, it is the state q' of machine M when, starting from the initial configuration q_0x, it reaches for the first time a configuration of the form xq'.

  If M halts on input x without ever moving past the end of x, then $f_x(begin)$ is the corresponding halting state q_A or q_R.

- $f_x(q)$ is the state q' of machine M when, starting from configuration $x_1, ..., q, x_n$ (that is, after "entering x from the right") it reaches for the first time a configuration of the form xq' (that is, the first time that it "re-exits" x). If M halts without every exiting x when started on configuration $x_1, ..., q, x_n$, then $f_x(q)$ is the corresponding halting state.

Now note what happens if $f_x = f_y$. For every string z, xz should be accepted if and only if yz is accepted: $f_x(begin) = f_y(begin)$ (if they both have halted by now then they've done so in the exact same way and we're done) and so they both enter z on the same state q. Since z is the same for both and they both are parsed starting at state q then they deterministically have to do the same thing and (if they halt then, again, we're done) so they will re-enter x on the same state q'. Now M may do different things on x and y but since $f_x(q') = f_y(q')$ they will either halt in the same way or re-exit in the same way and on the same state, so it will again transition identically on the shared z since they start on the same state (M essentially behaves the same on them). This argument can continue, so that any halting will be done in the exact same way and since a halt will happen (since M is a decider), xz is accepted if and only if yz is accepted (we could more rigorously induct over the number of times M crosses from x or y to z to show it's true for any number of crosses, which will be finite for given strings since halting must happen). And so x and y are indistinguishable by definition.

Therefore, the number of distinguishable strings has to be bounded by the number of possible functions. But there are only $|Q|^{|Q|+1}$ possible function by definition, which is finite. Thus, there can only be a finite number of distinguishable strings and the language decided by M must be regular by the Myhill-Nerode theorem.