L and NL

a On the homework, we used "R2TM"s (2-tape TMs where the input tape is read-only) to create a new notion of space complexity WSPACE which can more meaningfully talk about sublinear complexity. Sipser introduces a version of the model with a few additional tweaks, which I'll call the S2TM: "We provide a way for the machine to detect when the head is at the left-hand and right-hand ends of the input. The input head must remain on the portion of the tape containing the input" (p349). Briefly think about why the S2TM model for sublinear space might be preferable to R2TMs, and briefly consider how the new input-edge-detection feature could be reflected in the formal definition of a S2TM.

Solution:

Constraining the read-only head to the portion of its tape containing the input will allow us to more easily bound the number of possible configurations the TM can go through without repeating. It also allows us to be confident that the input tape can't be used as a secret extra source of unlimited storage - e.g. if we did not constrain the tape head, then it could store a counter based on how far right the head has gone beyond the input. (Though this counter might actually be useless since it can only be read destructively.)

For a normal TM, a built-in edge detection feature is not required because the tape is read-write, so we could always program a TM to just mark the edges of its input itself if we wanted. With a read-only input tape, we might instead keep track of the edges by storing a counter of how far right we've gone. But this solution stops working if we ever wanted to discuss sub-logarithmic space, since then even the counter is too large to store. One way to formally implement the edge-detection feature would be to modify the transition function signature from $Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{L, R, S\}^2$ to $Q \times \Gamma^2 \times \{leftEdge, rightEdge, noEdge\} \to Q \times \Gamma^2 \times \{L, R, S\}^2$, with the extra input being automatically supplied based on what edge the read-only head is at. *(Recall that multitape machines use* $\{L, R, S\}$ *instead of* $\{L, R\}$*)*

b From now on we will officially modify all our space complexity definitions (e.g. $SPACE(\cdot)$, $NSPACE(\cdot)$) to use S2TMs instead of TMs. (It'll be the same thing for linear space and above, and will help us discuss sublinear.) Define $\mathbf{L} = SPACE(\log n)$. For example, $\{0^n1^n \mid n \geq 0\} \in \mathbf{L}$, since we can decide it by checking that our input is of the form 0^*1^* and then counting the number of 0s and 1s to check for equality; the two counters take log space each. Show that $A_{DFA} \in \mathbf{L}$. *(Hint: an S2TM can store a pointer to a tape cell of its input in log space.)*

Solution: See solution (p360) to Sipser Exercise 8.5.

c Define $\mathbf{NL} = NSPACE(\log n)$. Show that $PATH = \{\langle G, s, t\rangle \mid G$ is a directed graph that has a (directed) path from s to $t\}$ is in **NL**. *(Note that you can not do things like store the whole path or "mark nodes we've visited", since there are linearly-many nodes and only log space to work with.)*

Solution: See Sipser Example 8.19 (p350)

d It is unknown whether **L**=**NL**. For P vs NP, we studied the issue using poly-time reductions ($\leq_P$) defined in terms of poly-time-computable functions, which we used to then define NP-completeness. Similarly, we will want log-space reductions ($\leq_L$) defined in terms of log-space-computable functions in order to define **NL**-completeness. Try to define "log-space-computable functions", keeping in mind that in logarithmic space you do not have space to store the input *or the output*. (You should create a new type of TM, in the same way that we defined S2TMs to handle the case where we can't store the input.) Sipser's definition appears on the next page; when you are done thinking here, compare your solution to that one and briefly think about whether any differences between your definitions seem important or just cosmetic.

Solution: See part (e). One difference I saw between some of your proposed definitions and Sipser's is that Sipser imposes the additional constraint that the write-only output tape head cannot move leftward. I'm not sure one actually needs *both* ways of making the tape write-only: if we make it write-only by having the transition function not take in a character from the third tape, then it probably doesn't matter if the head can move left since it won't be able to read anything it moves over. And if we make the head unable to move left, then it probably doesn't matter if the head can read, since there will never be anything to read other than the most recently written symbol or blanks.

e (Sipser Definition 8.21) A *log space transducer* is a TM with a read-only input tape, a write-only output tape, and a read/write work tape. The head on the output tape cannot move leftward, so it cannot read what it has written. The work tape may contain $O(\log n)$ symbols. A log space transducer M computes a function $f : \Sigma^* \to \Sigma^*$, where $f(w)$ is the string remaining on the output tape after M halts when it is started with w on its input tape. We call f a log space computable function. Language A is log space reducible to language B, written $A \leq_L B$, if A is mapping reducible to B by means of a log space computable function f.

Prove that if $A \leq_L B$ and $B \in L$, then $A \in L$. *(Note that given a log-space-computable function f, you can't actually just write "compute $f(w)$", since you don't have space to store the output. Hint: you are allowed to be* very *time-inefficient.)*

Solution: See Sipser Theorem 8.23 (p352)

f (if you have extra time) Recall that a directed graph is *strongly connected* if every two nodes are connected by a (directed) path in each direction. Let STRONGLY-CONNECTED $= \{\langle G \rangle \mid G$ is a strongly connected graph$\}$. Show that PATH $\leq_L$ STRONGLY-CONNECTED. (By Sipser Theorem 8.25 (p353) PATH is **NL**-complete, so this will show that STRONGLY-CONNECTED is **NL**-hard.)

Solution:

Deferred to homework