CS 181: HW9 TEJAS KAMTAM, 305748402

→ Should be reversed

3.10) We can simulate a write-once ~~TM~~ TM using a 2-tape TM which we know is equivalent to a reg. TM. On the first tape, we can have the input ~~sequence~~ string and every time we update a cell we want to ~~copy over~~ ~~mark~~ copy over the entire sequence to the right. To do so, we need to mark each cell that it has been copied over which can be indicated by the second tape. (this second tape acts more like as a paired set of cells than a true 2nd tape). For example consider an update $w_k \to v$, the resulting tape: ~~of the update~~:

~~Before: $w_1 \ldots w_k \ldots w_n$ | $\ldots$~~
~~After: $w_1 \ldots v \ldots w_n$~~

Before:

$w_1 \ldots w_k \ldots w_n$	$\ldots$		
	$\ldots$	$\ldots$	$\ldots$

After:

$w_1 \ldots w_k \ldots w_n$	$w_1 \ldots v \ldots w_n$		$\ldots$
$X_1 \ldots X_k \ldots X_n$	$\ldots$	$\ldots$	$\ldots$

3.12) We can show a left reset TM simulates a reg. TM so ~~the language of~~ left-reset TMs must recognize Turing-recognizable langs. The right movement works the same as a reg. TM so no need to change anything. But when the left-reset TM (M') resets, we want to show that it can behave the same as a TM. ~~To do so, we can add a marker $X \in \Gamma$ to the tape alphabet s.t. $\Gamma(M') = \Gamma \cup \{X\}$. Now, we can~~

To do so, we can make all of the cells contain a pair of symbols ~~when~~ instead of a single symbol so instead of, e.g., $q_7 \to b$ now $q_7 \to (b, \varepsilon)$. The second symbol can be empty (ε) or a mark (X). ~~When we reset,~~ Initially all cells will contain ε in the second symbol and when we reset we update $\varepsilon \to X$ and left reset, then shift all the first symbols to the

3.12 cont) ... Shift all of the first symbols to the right by 1 and ~~when we encounter~~ a cell then left-reset when we reach the end (⊔). Now continue right until the second symbol is X. When we find this cell we are now shifted left so this simulates the left shift of a reg. TM. Before continuing we have to turn $X \to \varepsilon$ to reset the cell state and continue w/ computation. E.g.:

Before: $\begin{pmatrix} w_1 \\ \varepsilon \end{pmatrix} \cdots \begin{pmatrix} w_k \\ \varepsilon \end{pmatrix} \cdots \begin{pmatrix} w_n \\ \varepsilon \end{pmatrix} \cdots$

After: $\begin{pmatrix} \varepsilon \\ \varepsilon \end{pmatrix} \begin{pmatrix} w_1 \\ \varepsilon \end{pmatrix} \cdots \begin{pmatrix} w_{k-1} \\ X \end{pmatrix} \begin{pmatrix} w_k \\ \varepsilon \end{pmatrix} \cdots \begin{pmatrix} w_n \\ \varepsilon \end{pmatrix} \cdots$

3.14) In class we showed that a queue can be made to be equivalent to 2 stacks; for example, every time we push, we push onto stack A, then to pull, we pop all of stack A and push to stack B then pop from stack B to complete ~~the~~ the pull. We have also discussed in class that an automaton w/ 2 stacks is equivalent to a TM. So, the queue automaton is equivalent to a TM so its language is Turing-recognizable, thus ~~if the~~ only Turing-recognizable languages can be recognized by the queue automaton.

3.16a) L_1, L_2 are Turing recognizable, prove ~~that~~ $L_1 \cup L_2$ is Turing-recognizable

We can construct 2-tapes s.t. both tapes contain the same input sequence (either at init or by copying tape 1 to 2). Then $\exists M_1, M_2$ s.t. $L(M_1) = L_1$ and $L(M_2) = L_2$, then run M_1 on tape 1 and M_2 on tape 2. One of these will halt ~~and accept as there exists a model that recognizes the input~~ b/c the models M_1, M_2 ~~can~~ will either accept or reject the input since the string is in L_1 or L_2. Thus if at least 1 model accepts then $w \in L_1 \cup L_2$, thus this TM recognizes the union

out of order

3.16d) Intersection

Similar to (3.16a), ~~as~~ but now only if both M_1 & M_2 accept then $w \in L_1 \cap L_2$. Thus this ~~model~~ TM recognizes the intersection of 2 Turing-recognizable langs.

3.16b) Concatenation

For ~~$w \in L_1 \circ$~~ ~~$w \in L_1 L_2$~~ $w \in L_1 \circ L_2$ (concatenation), we can construct a TM that recognizes $L_1 \circ L_2$ by splitting ~~w~~ $w \to xy$. Consider every possible split xy of w (for which there are finitely many ~~as~~ as inputs are finite) Then $\exists M_1, M_2$ s.t. $L(M_1) = L_1$ & $L(M_2) = L_2$ then run M_1 on x and M_2 on y and if both accept $\exists$ the TM accepts. Thus this TM recognizes $L_1 \circ L_2$ ~~and~~ $\therefore$ ~~that~~ TM-recognizable langs are closed under concat.

3.16c) For $w \in L_1^*$, split ~~w~~ w into substrings for which there are finitely many possibilities for the ~~finite~~ finite input. Then $\exists M_1$ s.t. $L(M_1) = L_1$, run M_1 on each of the substrings and ε, if M_1 accepts on all (substring are formatted as $w = x_1 x_2 \ldots x_n$ where $|x_i| > 0$) and ε, this TM recognizes ~~[illegible]~~ L_1^* $\therefore$ TM-recognizable langs. closed under Kleene-Star