Time and Space Complexity

This is an introduction to time and space complexity. A (multi-tape) TM M is said to be $t(n)$ time bounded if on every input of length n, it uses at most $t(n)$ steps. We will always assume that $t(n) \geq n$. For space bounds, this need not be the case; so we will have a read-only input tape and a (fixed) number of read-write tapes and measure only the space used on the read-write tapes as a function of the input length.

We will let DTIME($t(n)$) denote the class of languages that can be accpeted by $O(t(n))$ time bounded multi-tape TM's. Similarly, we define DSPACE($(s(n))$. We will only deal with "nice" time and space bound functions - $t(n), s(n)$, where nice will mean : non-decreasing, total recursive functions for both time and space bounds. In addition, for time bounds, $t(n)$, "nice" will also mean that we can compute the value of $t(n)$ (given n) in time at most $ct(n)$ for some constant $c > 0$ (indepent of n) by a 3-tape TM.

Note that we allow TM's with multiple tapes and also arbitrarily large (but finite) working alphabets. However, the following theorem shows that these enhancements do not add too much power.

Theorem Suppose a language $L \subseteq \{0,1\}^*$ is accepted in (nice) time bound $t(n)$ by a multi-tape TM M with an arbitrary fixed working alphabet. Then, for some constant $c_M > 0$ (independent of n, but depending on M), it is accepted by a $c_M(t(n))^2$ time bounded single tape TM with $\{0,1\}$ as its working alphabet.

Proof The single tape TM M' has the contents of each tape of the multi-tape TM M written on its tape. If the working alphabet of M has l letters in it, M' codes the i th letter by 0^i1 (say). Also, M' needs to remember the head position which it does by writing $0^{l+1}1$ just before the letter being read by the head. It is eacy to see that one step of M can be simulated by M' by running over its entire tape contents, which clearly takes time only $O(t(n))$ per step.

Hierarchy Theorem Suppose $t(n)$ is a nice time bound. Then, we have

$$\text{DTIME}(t(n)) \neq \text{DTIME}(n(t(n))^2).$$

Proof We will consider languages over $\{0,1\}$. so the input string will be a 0-1 string. But we will view the input string w as the encoding of a pair of strings x, y (in a standard way). We will construct a "digonalizing machine" M^*, which will be a 3 tape TM : On input w, our M^* does the following :

(i) Compute and store $\frac{1}{2}|w|(t(|w|))^2$; this can be done in time $\frac{1}{2}|w|(t(|w|))^2$. We will count each subsequent step of computation and stop M^* if it ever exceeds $\frac{1}{2}|w|(t(|w|))^2$ steps.

(ii) We decode the input w into x, y.

(iii) Then, M^* simulates the multi-tape (arbitrary working alphabet) TM described by x on the string w. Note that the simulated TM may have many more than 3 tapes; but we simulate it using just one of our 3 tapes as in the last Theorem. If ever the simulation terminates, we do the opposite of what the simulated TM did.

We claim that $L(M^*) \notin \text{DTIME}(t(n))$. To see this, suppose for contradiction that $L(M^*)$ was accpeted by a multi-tape $t(n)$ time bounded TM M. Then we have already seen that our simulation of M runs in time $c_M(t(n)^2)$, where c_M is a constant depending upon M alone. Now for large enough $n = |w|$, we will have $\frac{1}{2}n(t(n))^2 \geq c_M t(n)$ and so on a large enough string y, with x, y as input, where x is the description of M, our M^* will do the oppsite of M producing a contradiction.

Note : This is really using the following : we may view M^* as one machine which simulates any given $t(n)$ time bounded multi-tape TM (recall Universal Turing Machines). But the rub here is that M^* has afixed number of tapes, whereas the simulated TM's may have arbitrary number of tapes and working alphabet symbols.