Tape compression and completeness

Instructor: Prof. S. P. Pal
TA: Rahul Gokhale

Space and time are the two most important and commonly studied computational resources. We study space and time complexities of algorithms on TMs because TMs are very simple machines with finite state control and transition rules, and potentially infinite memory on tape(s). By Church's thesis, all other 'resonable' computational models are equivalent to TMs in computability, as well as, polynomially related in terms of time and space complexities.

Definition 1 *If for every input word of length n, TM M scans at most $S(n)$ cells on any writable storage or work tape, then M is said to be an $S(n)$ space-bounded TM, or of space complexity $S(n)$.*

Definition 2 *If for every input word of length n, TM M makes at most $T(n)$ moves before halting, then M is said to be a $T(n)$ time-bounded TM, or of time complexity $T(n)$.*

1 Tape compression: [Theoreom 12.1, HU79]

It is instructive to first get familiar with tape cell symbol manipulation to the extent that one machine M_2 is used to simulate another machine M_1 where each tape cell of M_2 encodes several (say r) cells of M_1. Using such an encoding $\lceil S(n)/r \rceil$[1] cells suffice in M_2 for $S(n)$ cells in M_1; $\lceil S(n)/r \rceil$ is upper bounded by $\left\lceil \frac{S(n)}{2/c} \right\rceil$ if $rc > 2$, for any choice $c > 0$. This bound is less than $\lceil cS(n)/2 \rceil$ and therefore bounded by $cS(n)$. Also if $S(n) < r$, then M_2 needs just one tape cell. So, M_2 can simulate M_1 in $cS(n)$ tape cells with compression factor r, where $rc > 2$.

See **[Lemma 12.1, Hu79]**. As we saw in class, the number of configurations of the simulated machine is $(n+2) \cdot s \cdot S(n) \cdot t^{S(n)}$; the simulating TM can use a $4st$ base counter with $S(n)$ cells to count at least upto such a number of configurations. It is worthwhile working out the details of the simulation where an additional counter cell is used every time a new tape cell is first visited by the simulated machine. So, we see that languages accepted by space bounded computations are also accepted within similar space bounds by a machine that halts.

Exercise 1 *Show that $4st$ raised to $S(n)$ exceeds the number of configurations of machine M_1 above.*

All this concerns a deterministic TM. What happens if the machines were non-deterministic?

[1] $\lceil x \rceil$ is the smallest integer larger than x.

2 Nondeterminstic machines and complexity classes

NFAs and DFAs that match in language acceptance capabilities have an exponential relationship in terms of the number of states. For the case of polynomial amount of writable memory as in the case of Turing machines running in polynomial time, nondeterminism gives rise to the famous **P**=**NP** question.

For definitions of **P** and **NP**, see [Papadimitriou 1994], henceforth called [P94]. (See page 181.)

Exercise 2 *Show that the decision problem of determining whether an n-vertex undirected graph has a vertex cover of size $\lceil n/3 \rceil$ is NP-complete.*

Exercise 3 *Show that the problem of determining whether the language $L(G)$ generated by a given context-free grammar G is the empty set, is in the class* ***P****. (See HU79). What happens for regular grammars? What is the input in these decision problems? [Note that regular grammars and context-free grammars may be nondeterministic.]*