CECS 419-519, Writing Assignment 9, Due 8:00 am, April 26th, 2024, Dr. Ebert

Directions

Make sure name is on all pages. Order pages (front and back) so that solutions are presented in their original numerical order. **Please no staples or folding of corners (your papers won't get lost). A paper clip is OK** Show all necessary work and substantiate all claims. Avoid plagiarism.

Problems

1. Recall the definition for what it means for a function $f : \mathcal{N} \to \mathcal{N}$ to be time constructible (respectively, space constructible). Prove that i) $f(n) = n^2$ is time constructible and ii) $f(n) = n$ is space constructible. Hint: since a Turing machine's δ-transition function can be programmed to satisfy the definition for a finite number of small inputs (say, $n = 0, 1, \ldots, k$, for any constant $k \geq 0$) focus your analysis on cases where n is "sufficiently large". (20 pts)

   Solution Part i) Suppose the tape contains $n > 0$ 1's. The idea is to read the input of 1's from left to right while maintaining a counter C whose most-significant bit (MSB) occupies the cell that is adjacent to the next 1 to be read and which counts the number of 1's that have already been read. To begin, the first 1 can be "converted" to a 1-bit counter C whose current count equals 1. Then C has one bit and it is adjacent to cell 2 which stores the next 1 to be read. Now suppose, for $k \geq 2$, the k th 1 is to be read and the C's MSB is located at cell $k-1$. Then the k th 1 is replaced with 0 and the head moves left to C's least-significant bit and 1 is added to the count while the head moves right. There are now two cases to consider.

   Case 1: As as a result of adding 1 to C, the previously written 0 has been overwritten with a 1, where 1 is an "overflow" bit. Then this bit now becomes C's new MSB and C's length increases by one. Moreover, C's MSB is adjacent to cell $k+1$ where the next 1 is to be read.

   Case 2: when adding 1 to C, there was no overlfow bit. In this case the bits of C must be shifted one place to the right so that C's MSB is adjacent to cell $k+1$, the location of the next 1 to be read.

   Thus, after all 1's have been read, C will hold the value n. Finally, since C's length is always at most $\lfloor \log n \rfloor + 1$, and shifting C's bits one place to the right requires $O(|C|)$ steps, it follows that the procedure requires $O(n \log n)$ steps.

   Solution Part ii) Since we may assume n satisfies $n > \log n$, we may allow counter C to remain in the first $\lfloor \log n \rfloor + 1$ cells and, for each 1 read by the head, the head moves back to C and adds 1. This procedure requires n cells and so is linear in n.

2. Review the statement of the Space Hierarchy Theorem, and let $t(n)$ be a function that satisfies $t(n) = o(s(n))$, meaning that the limit of the quotient $t(n)/s(n) \to 0$ as $n \to \infty$. Assume both $s(n)$ and $t(n)$ are space constructible. An instance of decision problem L is a pair $\langle M, w \rangle$, and the problem is to decide if M can accept w using at most $s(n)$ tape cells, where $n = |w|$. Then

L can be decided by a DTM $\hat{M}$ that uses at most $O(s(n))$ tape cells. Moreover, $\hat{M}$ works by simulating M on input w using the tape cells to the right of input $\langle M, w\rangle$. The goal of this exercise is to prove that L cannot be decided using $O(t(n))$ tape cells. Suppose by way of contradiction that there is a DTM M' that decides L using $O(t(n))$ tape cells. Similar to the proof of the Time Hierarchy Theorem, define a structured semiformal program Q that makes use of the `self` programmming construct and leads to a contradiction. Clearly explain why it creates a contradiction and defend your answer. (30 pts)

Solution.

> Input w.
>
> Mark tape cell $t(|w|)$. //Possible since $t(n)$ is space constructible.
>
> Simulate M' on input $\langle \texttt{self}, w\rangle$.
>
> If the simulation never reached cell $t(|w|)$ and $t(|w|) < s(|w|)$, then move to cell $s(|w|)$ and return $1 - M'(\langle \texttt{self}, w\rangle)$.
>
> Else return 0.

If the simulation never reaches cell $t(|w|)$ and $t(|w|) < s(|w|)$, then the computation continues and finishes by returning $1 - M'(\langle \texttt{self}, w\rangle)$. But then this is the result that will be returned by M when simulating Q on input w, and hence

$$M(\langle Q, w\rangle) \neq M'(\langle Q, w\rangle),$$

a contradiction.

3. In the proof of Theorem 2.9 of the Space Complexity lecture, a monotone grammar was constructed that is capable of deriving any word that is accepted by an arbitrary NTM $N = (Q, \Sigma, \Gamma, \delta, q_0, q_a)$ that writes on at most n cells, where n is the input length. Provide the third set of rules for this grammar, i.e. the ones that simulate N's tape head moving right. (10 pts)

   Solution.

   The following set of rules supports moving the tape head right. Suppose $(p, b, \mathrm{R}) \in \delta(q, a)$, then for each $c, e \in \Sigma$ and $d \in \Gamma$,

   $$(q, a, c)(d, e) \to (b, c)(p, d, e)$$
   $$(q, a, c)(d', e) \to (b, c)(p, d', e)$$

4. Design a DTM M that accepts all words of the form 0^n1^n, $n \geq 1$. Make sure that, whenever M accepts a word, that it halts in the accepting state at cell $2n$ (where the final 1 is located). Write its state diagram and provide an in-class demonstration of your machine using the simulator provided at

   `turingmachinesimulator.com`

   (20 pts)

   Solution. Solutions may vary.

5. Repeat Example 2.10 of the Space Complexity lecture, but now using your DTM from the previous problem when applied to input $w = 0011$. Provide a *complete unabridged* derivation via the monotone grammar defined in the proof of Theorem 2.9. (15 pts)

   Solution. Solutions may vary.

6. Recall that in the proof of Theorem 3.1 of the Space Complexity lecture, the functions $f(n)$ and $g(n)$ were defined in relation to a language $A \in \texttt{NSPACE}(n)$ and a monotone grammar G for which $A = L(G)$. Namely, $f(n)$ equals the number of length-n words that belong to A, and $g(n)$ equals the number of (not necessarily terminal) words of length $\leq n$ that can be derived by G. Provide a structured and semiformal nondeterministic linear-space program that computes $f(n)$ assuming $g(n)$ can be computed in nondeterministic linear-space (which it can via the `num_derived` program on page 24 of the lecture). Make sure that your program returns UNDEFINED in case the branch on which the computation is being performed may not have the correct value for $f(n)$. Hint: your program may call the `invert` program that was defined in the proof of Theorem 2.9. (15 pts)

   Solution.

   Name: `f`

   Inputs: i) $n \geq 0$, ii) monotone grammmar $G = (V, \Sigma, R, S)$.

   Output: the number of members of language A having length n.

   $m = g(n, G)$. //Both f and g have G as an implicit input.

   If $m =$ UNDEFINED, then return UNDEFINED.

   $\texttt{count}_g = 0$.

   $\texttt{count}_A = 0$.

   For each $i = 0, 1, \ldots, n$,

   For each $w \in (V \cup \Sigma)^i$ for which $\texttt{Invert}(w, G) = 1$,

   $\texttt{count}_g = \texttt{count}_g + 1$.

   If $i = n$ and $w \in \Sigma^i$, then $\texttt{count}_A = \texttt{count}_A + 1$.

   If $\texttt{count}_g \neq m$, then return UNDEFINED. //Some derivable words were missed on this branch.

   Return $\texttt{count}_A$.