Hierarchy Theorems

CS254

Chris Pollett

Oct 9, 2006.

Outline

- Complexity Classes
- The Hierarchy Theorem
- Relationships Between Classes

Complexity Classes

- Complexity classes are made up of several parameters.
  - Model of computation -- we are using k-tape machines
  - Mode -- we will consider deterministic and nondeterministic modes
  - Resource -- we are mainly interested in the resources of time and space.
  - Bound -- how many units of the given resource we are allowed to use. This is typically given by some function f.

Proper Complexity Functions

- What makes a good function for a resource bound?
  - We want to avoid weird functions like $f(n) = 0$ if n is even and is equal to 2^n if n is odd.

Def$^{\underline{n}}$ Let f be a function from the nonnegative integers to the nonnegative integers. We say *f is a proper complexity function* if f is nondecreasing and there is a deterministic TM M_f which when started with x on the input tape, runs for $t=O(|x| + f(|x|))$ steps using at most $O(f(|x|))$ space which outputs to an output tape the string $I^{f(|x|)}$. Here space is measured disregarding the input, and output. The input tape is assumed to be read-only and we assume we also have a dedicated, write-only output tape.

- For example, a constant c, n, [log n] are all proper complexity functions. One can show that is f and g are proper so are f+g, f*g, and 2^g.

Precise Machines

- One application of proper complexity classes is that we can get our machines to run for the same amount of time on all inputs of the same length.

Def$^{\underline{n}}$ A Turing machine M is ***precise***, if there are functions f and g, such that for every input x of length n and every computation of M on x, M halts after precisely f(n) steps and except for the input and output tape, all other tapes have used length precisely g(n).

Prop. Suppose (deterministic or nondeterministic) M decides L within time (or space) f(n), where f is a proper function. Then there is a precise machine M´ , ehich decides the same language in time (resp. space) O(f(n)).

Proof. We will only consider the time case, the space case is similar. First on input x, M´ computes the machine M_f for f using a new set of tapes. After this computation ends, M´ has $I^{f(|x|)}$ written on some tape. M´ rewinds this tape. So far the computation is deterministic. M´ then begins simulating M which may be deterministic or nondeterministic. At each step of simulating M, M´ advances the tape head on the tape with $I^{f(|x|)}$ written on it. M halts in at most f(n) steps. At which point, M´ remembers how M halted and continues down the tape with $I^{f(|x|)}$ on it until it gets to a blank. At this point it halts in the same way as M.

Specific Complexity Classes

- We will be interested in the complexity classes TIME(f), SPACE(f), NTIME(f), and NSPACE(f) usually for some proper function f.
- We will also be interested in the unions of such classes. For instance,

  $P = U_{j>0}TIME(n^j)$,

  $NP = U_{j>0}NTIME(n^j)$,

  $PSPACE = U_{j>0}SPACE(n^j)$,

  $NPSPACE = U_{j>0}NPSPACE(n^j)$,

  $E = U_{j>0}TIME(2^{j*n})$,

  $EXP = U_{j>0}TIME(2^{n\wedge j})$,
- Finally, we will consider the classes L=SPACE(log n) and NL=NSPACE(log n). For these classes, the input tape is read-only and space is measured in terms of squares used on the work tapes.
- Given a class of language C, the class of languages co-C are defined as $\{\overline{L} \mid L \text{ is in } C\}$. For example, co-r.e., co-NP, etc.

The Hierarchy Theorem

- We next turn to the question of when can we show a language is in TIME(f) but not in TIME(g)? Here g is a slower growing proper function than f.
- To do this we are going to look at clocked based variant of the halting problem. For $f(n) \geq n$, define:

  $H_f = \{ \langle M, x \rangle \mid M$ accepts input x after at most $f(|x|)$ steps$\}$

More on the Hierarchy Theorem

Lemma. $H_f \in TIME((f(n))^3)$.

Proof. We will give a variation of a universal machine to prove this. Let U_f have 4 tapes.

Let n= |<M,x>|. First, U_f use M_f to write an "alarm clock" amount of time $I^{f(n)}$ on the fourth tape. The fourth head is then rewound. Using the input <M,x>, the second tape is initialized to encode the start state s of M, <M> is copied to the third tape and converted into a 1-tape machine using our earlier simulation (this machine executes one step of the original machine in at most O(f(n)) steps and we could speed this to f(n)). The first tape is set up with just x on it. Doing all this, takes O(f(n)) time. The modified M is then simulated step by step. This involves:

- Comparing the current state, and current head position of tape 1 against the transition function of M on tape 3 until a match its found
- applying the appropriate change to the first tape and updating the current state on the second tape.
- advancing the alarm clock tape.

Simulating one step of the modified M take O(f(n)) steps on U_f. So simulating one step of the original takes $O(f(n)^2)$ steps on U_f. We simulate exactly f(n) steps of M by using our clock and determine if we have accepted by then. So the total time is $O(f(n)^3)$, which gives $H_f \in TIME((f(n))^3)$. as desired.

Still More on the Hierarchy Theorem

Lemma $H_f \notin$ TIME(f([n/2]).

Proof. Suppose M_{H_f} decides H_f in time f([n/2]). Let D_f be a diagonalizing machine that computes:

D_f(<M>) : if M_{H_f}(<M, M>) = "yes" then "no" else "yes".

This machine on input <M> runs in the same time as M_{H_f} on input <M,M>, that is, in time f([2n+1/2]) =f(n).

Does D_f accept its own description? Suppose D_f(<D_f>) = "yes". By our definition of D_f this is done in at most f(n) steps. This means M_{H_f}(< D_f, D_f >) = "no", which means < D_f, D_f > $\notin H_f$. This means D_f on input D_f did not accept in f(n) steps, contradiction our starting assumption. Starting with D_f(<D_f>) = "no" we can follow a similar chain of reasoning to get a contradiction.

Putting it all together

- Using the last two lemmas together one can show:

The Time Hierarchy Theorem. If $f(n) \geq n$ is a proper complexity function, then the class TIME(f(n)) is strictly contained in TIME($f(2n+1)^3$).

- This can be improved to TIME($f(n)\log^2 f(n)$).

Corollary. P is a proper subset of EXP.

- A similar technique as above can be used to show:

The Space Hierarchy Theorem. If f(n) is a proper function, then SPACE(f(n)) is a proper subset of SPACE(f(n)log f(n)).

Relationships Between Space and Time

- The following theorem collects together a lot of what we know about the relationships between time and space classes:

Theorem. Suppose f(n) is a proper complexity function. Then:

a) SPACE(f(n)) ⊆ NSPACE(f(n)) and TIME(f(n)) ⊆ NTIME(f(n)).

b) NTIME(f(n)) ⊆ SPACE(f(n)).

c) NSPACE(f(n)) ⊆ TIME($k^{\log n + f(n)}$)

Proof. (a) is trivial. To see (b) recall our machine to simulate a nondeterministic machine by a deterministic one, many lectures back could simulate up to f(|x|) steps using the exact same space as the nondeterministic machine except for one auxiliary tape that stores only string up to length f(|x|) coding the nondeterministic choices. We will prove (c) on the next slide.

More on Relationships Between Space and Time

For (c), let L be a langauge in NSPACE(f(n)) and let M decide it. We assume the input tape is read only and we can ignore the output tape. Therefore a configuration of a k-tape machine looks like (q, i, w_2,u_2,…, w_k,u_k). Notice this input tape is replace with an index i of the tape square we are reading. There are a total of at most

$$|K|*(n+1)*|\Sigma|^{(2k-2)|f(n)|}$$

configurations. i.e, $c^{\log n + f(n)}$ configurations for some c depending only on M. Given this space of configurations define a graph where we have an edge between two configuration C and C´ if C there is a one step transition from C to C´ according to M. So determining if x is in L is equivalent to checking if an accepting halt configuration is reachable from the start configuration of M on x. Since the reachability algorithm we had before is quadratic in the size of the graph, we can determine this in time $O((c^{\log n + f(n)})^2) = O(k^{\log n + f(n)})$ where $k=c^2$. Linear speed up then gives the result.

Corollary. L ⊆ NL ⊆ P ⊆ NP ⊆ PSPACE.