CSE547 Class 22

Jeremy Buhler

April 17, 2017

1 Automata... in... Spaaaaaace!

- So far, we've mostly focused on time complexity of TMs.
- We'll now think about space complexity – how many tape cells a TM uses for a given size of input.
- **Defn**: Let L be a language. We say that $L \in \text{DSPACE}(f(n))$ if L can be decided by a deterministic TM M that *reads* no more than $O(f(n))$ distinct cells of its tape to decide an input of size n.
- Note that M cannot write more cells than it reads, and that "skipping over" a cell without changing it still counts as reading.
- Note also that we don't specify the number of tapes for M, because we can simulate any number of tapes without changing the amount of space used by multiplexing the kth cells of all tapes into the kth cell of a single tape.
- **Defn**: Similarly, $L \in \text{NSPACE}(f(n))$ if L can be decided by a *nondeterministic* TM M that reads no more than $O(f(n))$ distinct cells of its tape to decide an input of size n.
- Analogous to time class P, we may define space classes

$$\text{PSPACE} = \bigcup_{k>0} \text{DSPACE}(n^k),$$

and

$$\text{NPSPACE} = \bigcup_{k>0} \text{NSPACE}(n^k).$$

How is space related to time?

- A TM can read at most one cell per move.
- Hence, if it makes at most $t(n)$ moves on inputs of size n, it uses space at most $t(n)$.
- More formally, if $L \in \text{DTIME}(t(n))$, then $L \in \text{DSPACE}(t(n))$.
- Now consider a TM M that uses at most $f(n)$ space for an input of size n.
- How many distinct configurations can M be in before it halts?

- (M cannot repeat a configuration, else it would loop forever.)
- M can have $|Q|$ possible states, $|\Gamma|$ possible values for each of the $f(n)$ cells it touches, and $f(n)$ possible head positions.
- Hence, total number of configurations is at most

$$|Q| \times f(n) \times |\Gamma|^{f(n)} = 2^{O(f(n))}.$$

- Conclude that M takes time at most exponential in the amount of space it uses!
- More formally, if $L \in \text{DSPACE}(f(n))$, then $L \in \text{DTIME}(2^{O(f(n))})$.
- Note that the converse does not hold – a TM that runs for time exponential in n may touch way more than polynomially many cells in n.

2 Relation of DSPACE and NSPACE

- We saw previously that the relationship between DTIME and NTIME is complicated.
- A deterministic TM can simulate a nondeterministic TM, but doing so in the "obvious" way blows up its running time exponentially!
- There is no non-obvious polynomial-time simulation unless $P = NP$.
- Is there a similarly fraught relationship between PSPACE and NPSPACE?
- Let $L \in NSPACE(f(n))$ be given, and suppose L is decided by an NTM N.
- Consider the space used by the simulation we studied previously.
- Recall that our deterministic simulator M carries out the computation of N for each possible sequence of nondeterministic choices to see if any sequence causes N to accept its input.
- As we saw above, a computation of N for one sequence of choices might take time $2^{O(f(n))}$.
- Moreover, every move of N might involve a nondeterministic choice!
- Hence, our simulator might have to write down a string of $2^{O(f(n))}$ choices to simulate one possible computation of N.
- Conclude that our naive simulation blows up the space of N by an exponential factor!

Does this mean that we're similarly clueless about the possibility of a polynomial-space simulation? **NO!**

- **Thm** (Savitch): For any $f(n) \geq n$, if $L \in DSPACE(f(n))$, then $L \in NSPACE(f^2(n))$.
- **Pf**: Let N be an NTM deciding L.
- Rather than simulate N directly on input w, we'll test indirectly whether there exists an accepting computation of N on w.

- First, we'll describe a TM-computable function, YIELDN, to determine whether an NTM can move from one configuration to another within a given time and space bound.
- Then, we'll show how to use YIELDN to compute what we want.

First, let's talk about the core function.

- Define a function YIELDN$^i(c, c', t)$ that returns true iff N can go from c to c' in at most t moves while using space at most i.
- If $t = 1$, YIELDN$^i(c, c', 1)$ returns true iff either $c = c'$ or $c \vdash c'$ according to N.
- For $t > 1$, let C^i be the set of all configurations of N using space at most i.
- Assume in what follows that t is a power of 2 (we only call YIELDN for powers of 2 in this construction.)
- Then YIELDN$^i(c, c', t)$ returns true iff there exists $c_m \in C^i$ for which
  - YIELDN$^i(c, c_m, t/2)$ returns true, and
  - YIELDN$^i(c_m, c', t/2)$ returns true.

We now build a deterministic TM M to decide if $w \in L$.

- Suppose that
  - N runs in space $f(n)$,
  - N runs for at most $2^{df(n)}$ steps for some constant d.
- Let $c_0(w)$ be the initial configuration of N on input w, and WLOG let c_a be its unique accepting configuration.
- (We can modify N without changing its space usage so that when it accepts, it erases its tape and halts in c_a.)
- Then N accepts w iff YIELDN$^{f(n)}(c_0(w), c_a, 2^{df(n)})$ returns true.
- But M cannot actually compute $f(n)$ – we don't know that this function is even computable by a TM!
- (However, M is allowed to depend on the constant d.)
- Instead, M will compute YIELDN^i for successively larger values of i until it gets an answer.

OK, what exactly does M do?

- On input w, M does the following for $i = 1, 2, \ldots$.
  - First, M computes YIELDN$^i(c_0(w), c_a, 2^{di})$.
  - If true, N accepts w in space at most i, so M accepts w.

  - Second, M computes YIELDN$^{i+1}(c_0(w), c', 2^{di})$ for each possible configuration c' of length $i+1$.
  - If false, then N neither accepts nor uses space more than i on input w, so it must reject w. Hence, M rejects w.
  - If neither test is conclusive, then M increments i and keeps going.

- Eventually, M will accept or reject w for some $i \leq f(n)$, since the second test definitely fails for $i = f(n) + 1$.
- Moreover, the space used by M is at most the space required to compute YIELDN$^{f(n)+1}(c, c', 2^{df(n)})$, since M can erase and reuse its main tape on each call to YIELDN.
- (M uses a second tape of size $O(n + \log f(n))$ to remember w and keep track of i in binary. If $f(n) \geq n$, then the first tape dominates the space usage.)

It remains to determine how much space M requires to compute YIELDN.

- As described above, we can compute YIELDN$^i(c, c', t)$ recursively.
- Suppose we implement the recursive calls using an explicit stack.
- Before each recursive call, we must push context (c, c', c_m, t) onto the stack, using space $O(i)$. (Note that $t = O(2^{di})$ and so requires only $O(i)$ space to store in binary.)
- In the base case, YIELDN^1 can be surely computed in space proportional to $|c| + |c'|$, just by comparing the two configurations and consulting the (finite) move function of N.
- Each configuration has size at most i, so base case uses $O(i)$ space.
- We need a total of $O(\log t) = O(i)$ levels of recursion, with (as shown) $O(i)$ space per level.
- Conclude that a depth-first recursive implementation needs only $O(i^2)$ space.
- i never exceeds $f(n) + 1$, so M's total space usage is only $O(f^2(n))$. QED

3 Conclusions

So what can we say about space complexity classes?

- Savitch's Theorem shows that PSPACE = NPSPACE, since nondeterminism can be simulated using only a polynomial blowup in space.
- We also know that $NP \subseteq$ NPSPACE, since any TM that runs in polynomial time uses at most polynomial space.
- Finally, let EXPTIME be the set of all languages decided by a deterministic TM that runs in time at most exponential in its input size.
- We know that PSPACE $\subseteq$ EXPTIME, since the time to run a TM is at most exponential in its space usage.

- To summarize, we have that

$$P \subseteq NP \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXPTIME}.$$

- It can be shown that $P \neq \text{EXPTIME}$, so at least one of the subset inclusions is proper.