CS256 Applied Theory of Computation

Complexity Classes II

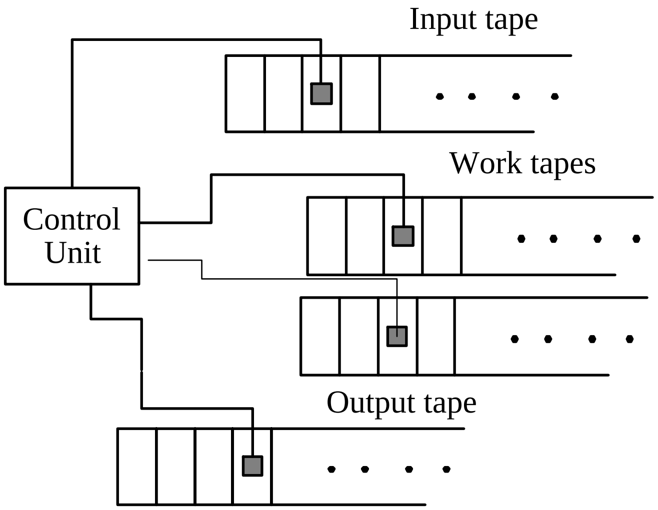
John E Savage

Overview

- Last lecture on time-bounded complexity classes
- Today we examine space-bounded complexity classes
- We prove *Savitch's Theorem*, i.e. NSPACE(r(n)), the set of languages recognized in non-deterministic space r(n), is contained in SPACE(r^2(n)).
- A consequence of Savitch's Theorem is that PSPACE = NPSPACE
- Nondeterminism doesn't increase the power of TMs when space used is polynomial length of the input!

Space-Bounded Computational Model

- The space-bounded multi-tape TM
- Input tape is read-only
- Output tape is write-only
- Space on input of length n is maximum number of cells used on worktapes



Space-Bounded Complexity Classes

Class	Space	By	Note
L	logarithmic	DTM	
NL	logarithmic	NDTM	$L \subseteq NL$
L^2	square log	DTM	
PSPACE	polynomial	DTM	
NPSPACE	polynomial	NDTM	$PSPACE \subseteq NPSPACE$

REACHABILITY

- REACHABILITY is used to show that PSPACE = NPSPACE.

  REACHABILITY

  Instance: A directed graph G = (V, E) and a pair of vertices u,v in V.

  Answer: "Yes" if there is a directed path in G from u to v.

REACHABILITY

- From transitive closure of adjacency matrix G we can decide REACHABILITY by depth first search (DFS). A RAM program for DFS can be implemented in $T=O(|E|)$ steps.
- T-step RAM program can be simulated on DTM in $O(T^3)$ steps, which implies that DFS can be realized in $O(|E|^3)$ steps.

REACHABILITY (cont.)

- Unfortunately, the space used for DFS on both machines is polynomial in |V| and |E|. Savitch's Theorem shows REACHABILITY can be realized in space $O(log^2 |V|)$, a big improvement.

- We use this fact to show PSPACE = NPSPACE

Savitch's Theorem

Theorem REACHABILITY is in space(log^2 n), n=|V|

Proof Let m=$\lceil log_2 n \rceil$. For vertices a, b in V & k ≤ m, let path(a,b,2^k) = 1 if there is a path between a and b of length ≤ 2^k and 0 otherwise. A "yes" instance (G,u,v) of reachability has path(u,v,2^m) = 1.

Because path(a,b,2^k) =1 if & only if exists vertex z such that path(a,z,2^{k-1}) = 1 & path(z,b,2^{k-1}) =1, we can evaluate path(a,b,2^k) recursively. Deterministic O(log^2 |V|)-space DTM for path(u,v,2^m) is now given.

Savitch's Theorem

Proof (cont.)

To compute path(a,b,2^k):

a) path(a,b,1) = 1 if (a,b) in E, 0 otherwise.

b) For z form prefix (a,z,k-1) & suffix (z,b,k-1).

c) Recursively evaluate prefix and suffix & compute

$$\text{path}(x,y,2^k) = \text{path}(x,z,2^{k-1}) \wedge \text{path}(z,y,2^{k-1})$$

d) Repeat until path(a,b,2^k)=1 or exhaust all z

Sketch of Savitch's Implementation

- Goal: Determine storage used on a DTM.
- Let DTM have one input tape, **two** work tapes.
- On input tape place u, v and edges E.
- First work tape is **stack** for activation records {(a,b,k)} (ARs). (a,b,k) associated with path(a,b,2^k). Put (u,v,m) on empty stack.
- Second work tape holds info on ARs

Sketch of Savitch's Implementation (cont.)

- Given a *complete* AR (a,b,k) and vertex z, we form *prefix* (a,z,k-1) and *suffix* (z,b,k-1).
- A complete AR (a,b,k) is *finished* if the value of path(a,b,2^k) is computed and *unfinished* if not.
- Initially (u,v,m) is complete and unfinished.
- To compute path(a,b,2^k), use stack to hold activation records during computation.

Sketch of Savitch's Implementation (cont.)

- To eval. complete, unfinished AR $\alpha = (a,b,k)$:
  - If k = 0, α has value 1 if & only if a=b or (a,b) in E
  - If k > 0, push (a,b,k), choose vertex z
  - * Evaluate prefix $\beta = (a,z,k-1)$ recursively.
  - If $\beta = 1$, evaluate suffix $\gamma = (z,b,k-1)$ recursively.
    - If $\gamma = 1$, set $\alpha = 1$. If $\gamma = 0$, choose next z; go to *.
  - If all z have been tried, set $\alpha = 0$.
  - If $\beta = 0$, choose next z & go to *. If no z left, $\alpha = 0$.
- To determine if current AR, (a,b,k), is a prefix or suffix, compare to previous AR on stack.
- Clearly, if path(u,v,2^m) =1, there will be values for z's so that (u,v,m) evaluates to 1.

Storage Used by Savitch's Algorithm

- The stack holds ARs of form (a,b,k). Second work tape holds an intermediate vertex z for each record (a,b,k) and value of path(a,b,2^k) when it has been computed.
- Deepest record in stack has k = m and k decreases by 1 moving to front of stack where k ≥ 0. Thus, stack has at most m = O(*log* n) records, n = |V| vertices. Thus, at most O(log^2 n) bits are needed for the stack.
- The second work tape holds a value of z and a value of path(a,b,2^k) for each record on stack. Thus, storage used is at worst O(log^2 n) bits. Q.E.D.

Extension to General Case

Corollary Let r(n) be proper and $r(n) = \Omega(\mathit{log}\ n)$. Then, $\mathrm{NSPACE}(r(n)) \subseteq \mathrm{SPACE}(r^2(n))$.

Proof Let TM M have input, output and s work tapes. Its **configuration** is tuple $(p, h_1, \ldots, h_{s+1}, \mathbf{x}_1, \ldots, \mathbf{x}_{s+1})$ where p is TM state, h_j is the position of the head on the *jth* tape, and $\mathbf{x}_j$ are non-blank symbols on the *jth* tape. (We assume that the TM cannot write blanks.) From a current configuration of a TM the next configuration can be computed knowing the next-state/output table of the TM.

Extension to General Case

- **Proof (cont.)**

  Configuration graph G(M,**w**) for the TM M on input **w** has a vertex for each reachable configuration and edge between two vertices if one can be reached from the other. Note that a vertex has out degree of more than one if the configuration is nondeterministic.

Extension to General Case

Let M recognize L in NSPACE(r(n)). For input **w** of length n, M has at most $C=|Q|(nr(n)^s)c^{sr(n)}$ configurations because the control unit has at most |Q| states, the input head can be in one of n positions, the heads on each of the s work tapes can be in at most r(n) positions, and each work tape can assume at most $c^{r(n)}$ values, for some c. Thus, the configuration graph G(M,**w**) associated with M on input **w** has C vertices and at most C^2 edges.

Extension to General Case

- Now use Savitch's algorithm to solve reachability problem on G(M,**w**). Recall that it stores information on m = $\lceil log^2 N \rceil$ activation records where N is the number of vertices in the graph. The amount of space used is m times the maximum amount of space used to record an activation record.

Extension to General Case

- Describe vertices in the configuration graph using configurations. On the input tape write **w** and the description of M. This information is sufficient to allow for the determination of vertices that are adjacent to the current vertex.
- Because the no. N of configurations is $O(k^{\log n + r(n)})$, O(r(n)) storage suffices to hold one activation record. Also, the value of $m = \lceil \log^2 N \rceil$ is of the same order. The result follows. Q.E.D.