18.404 Recitation 9

Nov 6, 2020
Today’s Topics

● Review: Savitch’s Theorem
● Review: PSPACE reduction
● Review: TQBF is PSPACE-Complete
● Prove: $GG \in \text{PSPACE}$
● Prove: $NL \subseteq \text{SPACE}(\log^2(n))$
● Review: $NL \subseteq \text{P}$
● BIPARTITE $\in \text{coNL}$
Review: Savitch’s Theorem

Conclusion: PSPACE = NPSPACE

Proof: Convert NPSPACE NTM to PSPACE TM by only using square more space

Idea:

- Use $\text{LADDER}_{\text{DFA}} \in \text{PSPACE}$ construction where “words” in the LADDER are actually computation histories.
- Test for LADDER from start to accept configuration
Review: Savitch’s Theorem (cont.)

Proof: Deterministic TM $M$ simulating NTM $N$ which uses $f(n)$ space

$M = “On$ input $<c_i, c_j, b>$$

1. If $b = 1$, accept if $c_i \rightarrow c_j$ is a valid state transition for NTM $N$
2. For $b > 1$, repeat for all configs $c_{mid}$ that use $f(n)$ space
   a. Recursively test $c_i \rightarrow_{b/2} c_{mid}$, $c_{mid} \rightarrow_{b/2} c_j$
   b. Accept if both accept
3. Reject if all fail”

Test if $N$ accepts $w$ by testing $<c_{start}, c_{end}, t>$ where $t = \#$ configs $= |Q| \cdot f(n) \cdot |\Sigma|^{f(n)}$
Review: Savitch’s Theorem (Space Analysis)

Recall: $M = \text{“On input } \langle c_i, c_j, b \rangle$

1. If $b = 1$, accept if $c_i \rightarrow c_j$ is a valid state transition for NTM $N$
2. For $b > 1$, repeat for all configs $c_{\text{mid}}$ that use $f(n)$ space
   a. Recursively test $c_i \rightarrow_{b/2} c_{\text{mid}}$, $c_{\text{mid}} \rightarrow_{b/2} c_j$
   b. Accept if both accept
3. Reject if all fail”

Each recursion level stores 1 config = $O(f(n))$ space.

Number of levels = $\log(t) = O(f(n))$

Total: $O(f^2(n))$ space
Review: PSPACE Reduction

Definition: B is PSPACE-Complete if

1. $B \in \text{PSPACE}$
2. For all $A \in \text{PSPACE}$, $A \leq_p B$

Note: Reduction uses $p$ for polynomial time reduction machine
Review: TQBF is PSPACE-Complete

Goal: Compute if M on w accepts where:

- M is a TM which runs in PSPACE
- Reduction in TQBF uses polynomial time (aka $\leq_p$)

Note: Many of the initial attempts presented in class were not reducible in polynomial time. Usually there was an exponential blow up somewhere.
Use Cook-Levin transition boolean formula: $\varphi_{ci,cj,1}$ is well-defined

Reduction: (Recursively defined)

$$\varphi_{Ci,Cj,b} = \exists c_{mid} \left[ \forall (c_g,c_h) \in \{ (c_i,c_{mid}), (c_{mid}, c_j) \} \left[ \varphi_{Cg,Ch,b/2} \right] \right]$$

The $\varphi_{Cg,Ch,b/2}$ eventually recurses down to $\varphi_{Ci,Cj,1}$ which is well-defined

So: $\varphi_{M,w} = \varphi_{C\text{start},C\text{accept},t}$ where $t = d^{f(n)}$ (# configs before repeat)

Size Analysis: Each recursive level adds $O(f(n))$. #levels = $\log(d^{f(n)}) = O(f(n))$

Size is $O(f^2(n))$
Review: TQBF is PSPACE-Complete (Space Analysis)

Recall:

\[ \varphi_{C_i,C_j,b} = \exists C_{mid} \left[ \forall (c_g,c_h) \in \{(c_i,C_{mid}), (C_{mid}, c_j)\} \left[ \varphi_{c_g,c_h,b/2} \right] \right] \]

Size Analysis:

- Each recursive level adds constant number of configs to QBF: \(O(f(n))\)
- \(\text{#levels} = \log(d^{f(n)}) = O(f(n))\)

So: Size is \(O(f^2(n))\)
Prove: \( GG \in \text{PSPACE} \)

Idea: Develop polynomial-space recursive algorithm determining which player has a winning strategy

Proof: \( M = \text{“On input } <G, n_{\text{start}} > \text{”} \)

1. If \( n_{\text{start}} \) has no outgoing edges, \textit{reject} since no available move (signalling loss)
2. Remember list of nodes \([n_1, ..., n_i]\) reachable from \( n_{\text{start}} \) through a single edge
3. Remove \( n_{\text{start}} \) and all edges connecting it to form graph \( G' \)
4. For every \( n_j \in [n_1, ..., n_i] \), call \( M(G', n_j) \) (signalling the moves of the opponent)
5. If \text{any} call return \textit{accept}, means that opponent can always win, so we lose and therefore \textit{reject}. Otherwise \textit{accept} since we have a winning path.
Prove: \( \text{NL} \subseteq \text{SPACE}(\log^2(n)) \)

Same proof using Savitch’s Theorem

Proof: \( M = \text{“On input } <c_i, c_j, b> \text{”} \)

1. If \( b = 1 \), accept if \( c_i \rightarrow c_j \) is a valid state transition for NTM \( N \)
2. For \( b > 1 \), repeat for all configs \( c_{mid} \) that use \( \log(n) \) space
   a. Recursively test \( c_i \rightarrow_{b/2} c_{mid}, c_{mid} \rightarrow_{b/2} c_j \)
   b. Accept if both accept
3. Reject if all fail”
Prove: $NL \subseteq SPACE(\log^2(n))$ (Space Analysis.)

Recall: $M = \text{"On input } <c_i,c_j,b>\text{"}$

1. If $b = 1$, accept if $c_i \rightarrow c_j$ is a valid state transition for NTM $N$
2. For $b > 1$, repeat for all configs $c_{mid}$ that use $\log(n)$ space
   a. Recursively test $c_i \rightarrow_{b/2} c_{mid}$, $c_{mid} \rightarrow_{b/2} c_j$
   b. Accept if both accept
3. Reject if all fail$

Each recursion level stores 1 config $= O(\log(n))$ space.

Number of levels $= \log(t) = O(\log(n))$

Total: $O(\log^2(n))$ space
Review: $\text{NL} \subseteq \text{P}$

Define a configuration graph $G_{M,w}$ for $M$ on $w$ which has:

- nodes for all configurations $M$ on $w$
- edges for all valid transitions $c_i \rightarrow c_j$

Utilize TM $T$ deciding $\text{PATH} \in \text{P}$

Run $T$ on $<G_{M,w}, \text{c}_{\text{start}}, \text{c}_{\text{accept}}>$
Review: NL ⊆ P (Time Analysis)

Recall: Graph $G_{M,w}$ has:

- nodes for all configurations $M$ on $w$
- edges for all valid transitions $c_i \rightarrow c_j$

Theorem: Constructing $G_{M,w}$ can be done in polynomial time

Proof: Configurations space for NL is log(n), therefore at most $\log(df(n)) = f(n)$ steps for all possible executions. Which is polynomial in time

Also: PATH is also in P
BIPARTITE $\in$ coNL

Definition: An undirected graph is bipartite if:

- the nodes of the graph can be split into two groupings
- edges only span groupings
Lemma: A graph is bipartite if $G$ does not contain a cycle of odd number nodes.

Backward direction: In a cycle, the nodes of the same parity (even/odd) belong to the same grouping.

Forward direction: In order to form a cycle, need to leave grouping and come back to origin grouping. This forces an even parity always.
BIPARTITE ∈ coNL (cont.)

In coNL means NOT-BIPARTITE in NL. Define NL TM M for NOT-BIPARTITE

M = “On input <G>:

1. Nondet. guess node u, and remember it
2. Remember prev = u
3. For i = 1 ... #nodes:
   a. Nondet. guess node v
   b. If edge between prev and v does not exist, reject
   c. If v = u and i is odd, accept
   d. If edge exists, set prev = v
4. If loop ends without accepting, reject”
**BIPARTITE \( \in \text{coNL} \) (Space Analysis)**

Recall: M = “On input \(<G>\):

1. Nondet. guess node \( u \), and remember it
2. Remember \( \text{prev} = u \)
3. For \( i = 1 \ldots \#\text{nodes} \):
   a. Nondet. guess node \( v \)
   b. If edge between \( \text{prev} \) and \( v \) does not exist, reject
   c. If \( v = u \) and \( i \) is odd, accept
   d. If edge exists, set \( \text{prev} = v \)
4. If loop ends without accepting, reject”

Only remember \( u,v, \) and \( \text{prev} \) -- constant space. Remember counter \( i \) which is \( \log(n) \)