Theory of Computing
Space Complexity

Ming-Hsien Tsai

Department of Information Management
National Taiwan University

Spring 2019

(original created by Bow-Yaw Wang)

Space Complexity

Definition 1

Let M be a TM that halts on all inputs. The space complexity of M is $f : \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum number of tape cells that M scans on any input of length n.
If the space complexity of M is $f(n)$, we say M runs in space $f(n)$.

Definition 2

If N is an NTM wherein all branches of its computation halts on all inputs. The space complexity of N is $f : \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum number of tape cells that N scans on any branch of its computation for any input of length n.
If the space complexity of N is $f(n)$, we say N runs in space $f(n)$.

Space Complexity Classes

Definition 3

Let $f : \mathbb{N} \to \mathbb{R}^+$. The space complexity classes, $SPACE(f(n))$ and $NSPACE(f(n))$, are

$$\begin{aligned} SPACE(f(n)) &= \{L : L \text{ is decided by an } O(f(n)) \text{ space TM}\} \\ NSPACE(f(n)) &= \{L : L \text{ is decided by an } O(f(n)) \text{ space NTM}\} \end{aligned}$$

$SAT \in SPACE(n)$

Example 4

Give a TM that decides SAT in space $O(n)$.

Proof.

Consider
M_1 = "On input $\langle \phi \rangle$ where ϕ is a Boolean formula:

1. For each truth assignment to $x_1, x_2, \ldots, x_m$ of ϕ, do
   1. Evaluate ϕ on the truth assignment.
2. If ϕ ever eavluates to 1, accept; otherwise, reject."

M_1 runs in space $O(n)$ since it only needs to store the current truth assignment for m variables and $m \in O(n)$. □

Universality of NFA's

- Consider $ALL_{\text{NFA}} = \{\langle A \rangle : A \text{ is an NFA and } L(A) = \Sigma^*\}$.
  - ALL_{NFA} is not known to be in NP or in $coNP$.

Example 5

Show $ALL_{\text{NFA}} \in coNSPACE(n)$.

Proof.

Consider

$N =$ "On input $\langle A \rangle$ where A is an NFA with q states:

1. Place a marker on the start state of A.
2. Repeat 2^q times:
   1. Nondeterministically select an input symbol a and simulate A on a by changing (or adding) positions of the markers on A's states.
3. If a marker is ever place on an accept state, reject; otherwise, accept."

Observe that if A rejects any string, it rejects a string of length at most 2^q. Hence N decides $\overline{ALL_{\text{NFA}}}$. Moreover, N only needs to store locations of markers and the loop counter. N runs in space $O(n)$. □

Savitch's Theorem

Theorem 6 (Savitch)

For $f : \mathbb{N} \to \mathbb{R}^+$ *with* $f(n) \geq n$, $NSPACE(f(n)) \subseteq SPACE(f^2(n))$.

Proof.

Let N be an NTM deciding A in space $f(n)$. Assume N has a unique accepting configuration c_{accept} (how?). We construct a TM M deciding A in space $O(f^2(n))$. Let w be an input to N, c_1, c_2 configurations of N on w, and $t \in \mathbb{N}$. Consider
$CANYIELD$ = "On input $c_1, c_2,$ and t:

1. If $t = 1$, test whether $c_1 = c_2$, or c_1 yields c_2 in N. If either succeeds, accept; otherwise, reject.
2. If $t > 1$, for each configuration c_m of N on w do
    1. Run $CANYIELD(c_1, c_m, \frac{t}{2})$.
    2. Run $CANYIELD(c_m, c_2, \frac{t}{2})$.
    3. If both accept, accept.
3. Reject."

Observe that $CANYIELD$ needs to store the step number, c_1, c_2, and t for recursion.

Savitch's Theorem

Proof (cont'd).

We select a constant d so that N has at most $2^{df(n)}$ configurations where $n = |w|$.
M = "On input w:

1. Run $CANYIELD(c_{\text{start}}, c_{\text{accept}}, 2^{df(n)})$."

Since $t = 2^{df(n)}$, the depth of recusion is $O(\lg 2^{df(n)}) = O(f(n))$. Moreover, *CANYIELD* can store its step number, c_1, c_2, t in space $O(f(n))$. Thus M runs in space $O(f(n) \times f(n)) = O(f^2(n))$.
A technical problem for M is to compute $f(n)$ in space $O(f(n))$. This can be avoided as follows. Instead of computing $f(n)$, M tries $f(n) = 1, 2, 3, \ldots$. For each $f(n) = i$, M calls *CANYIELD* as before but also checks if N reaches a configuration of length $i + 1$ from c_{start}. If N reaches c_{accept}, M accepts as before. If N reaches a configuration of length $i + 1$ but fails to reach c_{accept}, M continues with $f(n) = i + 1$. Otherwise, all configurations of N have length $\leq f(n)$. N still fails to reach c_{accept} in $2^{df(n)}$ time. Hence M rejects. □

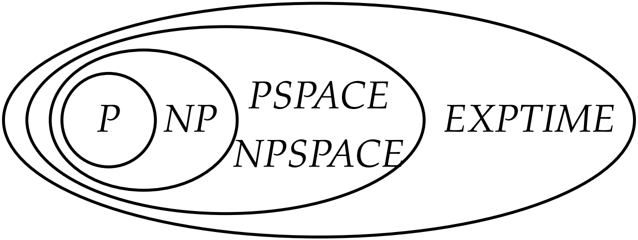
Definition 7

PSPACE is the class of languages decidable by TM's in polynomial space. That is,

$$PSPACE = \bigcup_k SPACE(n^k).$$

- Consider the class of langauges decidable by NTM's in polynomial space $NPSPACE = \bigcup_k NSPACE(n^k)$.
- By Savitch's Theorem, $NSPACE(n^k) \subseteq SPACE(n^{2k})$. Clearly, $SPACE(n^k) \subseteq NSPACE(n^k)$. Hence $NPSPACE = PSPACE$.
- Recall $SAT \in SPACE(n)$ and $ALL_{\text{NFA}} \in coNSPACE(n)$. By Savitch's Theorem, $\overline{ALL_{\text{NFA}}} \in NSPACE(n) \subseteq SPACE(n^2)$. Hence $ALL_{\text{NFA}} \in SPACE(n^2)$ (why?). $SAT, ALL_{\text{NFA}} \in PSPACE$.

- $P \subseteq PSPACE$
  - A TM running in time $t(n)$ uses space $t(n)$ (provided $t(n) \geq n$).
- Similarly, $NP \subseteq NPSPACE$ and thus $NP \subseteq PSPACE$.
- $PSPACE \subseteq EXPTIME = \cup_k TIME(2^{n^k})$
  - A TM running in space $f(n)$ has at most $f(n)2^{O(f(n))}$ different configurations (provided $f(n) \geq n$).
    - ★ A configuration contains the current state, the location of tape head, and the tape contents.
- In summary, $P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$.
  - We will show $P \neq EXPTIME$.



PSPACE-Completeness

Definition 8

A language B is <u>*PSPACE*-complete</u> if it satisfies

- $B \in PSPACE$; and
- $A \leq_P B$ for every $A \in PSPACE$.

If B only satisfies the second condition, we say it is <u>*PSPACE*-hard</u>.

- We do not define "polynomial space reduction" nor use it.
- Intuitively, a complete problem is most difficult in the class.
- If we can solve a complete problem, we can solve all problems in the same class easily.
- Polynomial space reduction is not easy at all.
  - Recall $SAT \in SPACE(n)$.

TQBF

- Recall the universal quantifier $\forall$ and the existential quantifier $\exists$.
- When we use quantifiers, we should specify a universe.
  - $\forall x \exists y[x < y \land y < x + 1]$ is false if $\mathbb{Z}$ is the universe.
  - $\forall x \exists y[x < y \land y < x + 1]$ is true if $\mathbb{Q}$ is the universe.
- A quantified Boolean formula is a quantified Boolean formula over the universe $\mathbb{B}$.
- Any formula with quantifiers can be converted to a formula begins with quantifiers.
  - $\forall x[x \geq 0 \implies \exists y[y^2 = x]]$ is equivalent to $\forall x \exists y[x \geq 0 \implies y^2 = x]$.
  - This is called prenex normal form.
- We always consider formulae in prenex normal form.
- If all variables are quantified in a formula, we say the formula is fully quantified (or a sentence).
- Consider

$$TQBF = \{\langle \phi \rangle : \phi \text{ is a true fully quantified Boolean formula}\}.$$

TQBF is *PSPACE*-Complete

Theorem 9

TQBF is PSPACE-complete.

Proof.

We first show $TQBF \in PSPACE$. Consider
$T =$ "On input $\langle \phi \rangle$ where ϕ is a fully quantified Boolean formula:

1. If ϕ has no quantifier, it is a Boolean formula without variables. If ϕ evaluates to 1, accept; otherwise, reject.
2. If ϕ is $\exists x \psi$, call T recursively on $\psi[x \mapsto 0]$ and $\psi[x \mapsto 1]$. If T accepts either, accept; otherwise, reject.
3. If ϕ is $\forall x \psi$, call T recursively on $\psi[x \mapsto 0]$ and $\psi[x \mapsto 1]$. If T accepts both, accept; otherwise, reject.

The depth of recursion is the number of variables. At each level, T needs to store the value of one variable. Hence T runs in space $O(n)$.

TQBF is *PSPACE*-Complete

Proof (cont'd).

Let M be a TM deciding A in space n^k. For any string w, we construct a quantified Boolean formula ϕ such that M accepts w if and only if ϕ is true. More precisely, let c_1, c_2 be collections of variables representing two configurations, and $t > 0$, we construct a formula $\phi_{c_1,c_2,t}$ such that $\phi_{c_1,c_2,t} \wedge c_1 = \mathtt{c}_1 \wedge c_2 = \mathtt{c}_2$ is true if and only if M can go from the configuration $\mathtt{c}_1$ to the configuration $\mathtt{c}_2$ in $\leq t$ steps.

To construct $\phi_{c_1,c_2,1}$, we check if $c_1 = c_2$, or the configuration represented by c_1 yields the configuration represented by c_2 in M. We use the technique in the proof of Cook-Levin Theorem. That is, we construct a Boolean formula stating that all windows on the rows c_1, c_2 are valid. Observe that $|\phi_{c_1,c_2,1}| \in O(n^k)$. For $t > 1$, let

$$\phi_{c_1,c_2,t} = \exists m \forall c_3 \forall c_4 \left[((c_3 = c_1 \wedge c_4 = m) \vee (c_3 = m \wedge c_4 = c_2)) \implies \phi_{c_3,c_4,\frac{t}{2}} \right]$$

Note that $|\phi_{c_1,c_2,t}| = \gamma n^k + |\phi_{c_3,c_4,\frac{t}{2}}|$ for some constant γ.

Assume M has a unique accepting configuration c_{accept}. Choose a constant d so that M has at most 2^{dn^k} configurations on w. Then $\phi_{c_{\text{start}},c_{\text{accept}},2^{dn^k}}$ is true if and only if M accepts w. Moreover, the depth of recursion is $O(\lg 2^{dn^k}) = O(n^k)$. Each level increases the size of $\phi_{c_1,c_2,t}$ by $O(n^k)$. Hence $|\phi_{c_{\text{start}},c_{\text{accept}},2^{dn^k}}| \in O(n^{2k})$. $\square$

TQBF is *PSPACE*-Complete

- Do we really need quantified Boolean formulae?
- For $t > 1$, consider

$$\phi_{c_1,c_2,t} = \exists m[\phi_{c_1,m,\frac{t}{2}} \wedge \phi_{m,c_2,\frac{t}{2}}].$$

- Recall that $\phi_{c_1,c_2,1}$ is an unquantified Boolean formula.
- We can construct an unquantified formula $\Phi_{c_1,c_2,t}$ such that $\langle\phi_{c_1,c_2,t}\rangle \in TQBF$ if and only if $\langle\Phi_{c_1,c_2,t}\rangle \in SAT$.
- Hence $PSPACE \subseteq NP$?!
- Note that $|\phi_{c_1,c_2,t}| \geq 2|\phi_{c_1,c_2,\frac{t}{2}}|$. $|\phi_{c_1,c_2,2^{dn^k}}|$ is in fact of size $O(2^{n^k})$.
- Quantifiers allow us to "reuse" subformula!

TQBF is *PSPACE*-Complete

- Do we really need quantified Boolean formulae?
- For $t > 1$, consider

$$\phi_{c_1,c_2,t} = \exists m[\phi_{c_1,m,\frac{t}{2}} \wedge \phi_{m,c_2,\frac{t}{2}}].$$

- Recall that $\phi_{c_1,c_2,1}$ is an unquantified Boolean formula.
- We can construct an unquantified formula $\Phi_{c_1,c_2,t}$ such that $\langle\phi_{c_1,c_2,t}\rangle \in TQBF$ if and only if $\langle\Phi_{c_1,c_2,t}\rangle \in SAT$.
- Hence $PSPACE \subseteq NP$?!
- Note that $|\phi_{c_1,c_2,t}| \geq 2|\phi_{c_1,c_2,\frac{t}{2}}|$. $|\phi_{c_1,c_2,2^{dn^k}}|$ is in fact of size $O(2^{n^k})$.
- Quantifiers allow us to "reuse" subformula!

Formula Games

- Let $\phi = \exists x_1 \forall x_2 \exists x_3 \cdots \mathsf{Q} x_k [\psi]$ (Q denotes $\exists$ or $\forall$) be a quantified Boolean formula in prenex normal form.
- In a formula game, Player A and Player E take turns selecting values for $x_1, x_2, \ldots, x_k$.
  - Player A selects values of $\forall$-quantified variables;
  - Player E selects values of $\exists$-quantified variables.
- The order of play is determinied by ϕ.
- At the end of play, all variables have their values.
  - Player E wins if ψ evaluates to 1;
  - Player A wins if ψ evaluates to 0.
- A player has a winning strategy for the game associated with ϕ if the player wins when both sides play optimally.

Formula Games

Example 10

Let $\phi_1 = \exists x_1 \forall x_2 \exists x_3 [(x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})]$. Show Player E has a winning strategy.

Proof.

Consider the following strategy for Player E

1. Player E starts by selecting $x_1 = 1$.
2. Player E selects the value of x_3 as follows.
   1. If Player A selects $x_2 = 0$, Player E selects $x_3 = 1$;
   2. If Player A selects $x_2 = 1$, Player E selects $x_3 = 0$.

It is easy to verify that Player E always wins. $\square$

- Consider

$$FORMULAGAME = \{\langle \phi \rangle : \text{Player E has a winning strategy in the formula game associated with } \phi\}.$$

FORMULAGAME is *PSPACE*-Complete

Theorem 11

FORMULAGAME is PSPACE-complete.

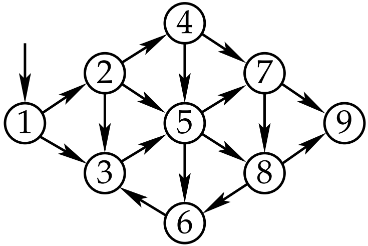
Proof.

The formula $\phi = \exists x_1 \forall x_2 \exists x_3 \cdots [\psi]$ is true if there is a value of x_1 such that no matter what value of x_2 is $\exists x_3 \cdots [\psi]$ is true. This is exactly when Player E has a winning strategy. □

Generalized Geography

- In generalized geography, a directed graph G with a designated start node b (a path of length 0) are given.
- Start by Player I. Player I and II takes turns to move.
  - At each move, a player selects a neighboring node that form a simple path in the graph.
- The first player fails to extend the path loses the game.
- Consider

$$GG = \{\langle G, b\rangle : \text{ Player I has a winning strategy for the generalize geography game played on } G \text{ starting at node } b\}$$



Player I wins by selecting node 3

GG is *PSPACE*-Complete

Theorem 12

GG is PSPACE-complete.

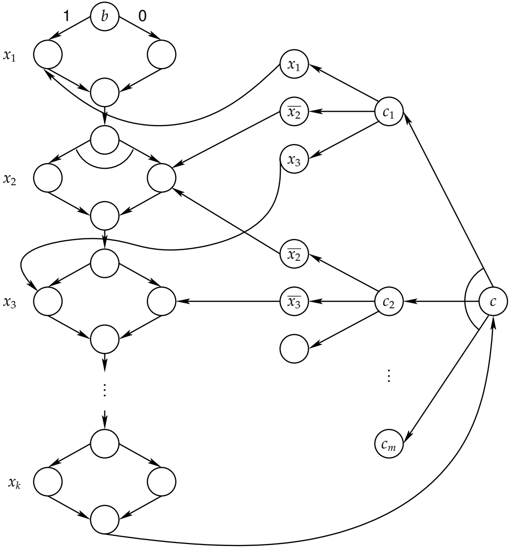
Proof.

We first show $GG \in PSPACE$. Consider
$M =$ "On input $\langle G, b \rangle$ where G is a directed graph and b a node of G:

1. If b has outdegree 0, reject.
2. Remove b and all connected edges to obtain G'.
3. For each nodes $b_1, b_2, \ldots, b_k$ pointed by b in G, call M on $\langle G', b_i \rangle$ recursively.
4. If M accepts $\langle G', b_i \rangle$ for all i, reject. Otherwise, accept."

The depth of recursion is the number of nodes in G. At each level, M stores a node. Hence M runs in space $O(n)$.
We now give a polynomial time reduction of *TQBF* to *GG*. Let $\phi = \exists x_1 \forall x_2 \exists x_3 \cdots \exists x_k [\psi]$ be a quantified Boolean formula where ψ is in 3CNF. (If ϕ is not alternating or ends with an $\exists$-quantifier, add dummy variables.)

GG is $PSPACE$-Complete



$$\phi = \exists x_1 \forall x_2 \cdots \exists x_k [(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee \cdots) \wedge \cdots]$$

GG is *PSPACE*-Complete

Proof.

We construct G as follows.

- For each variable x_i, a variable gadget consists of a diamond. The left branch denotes the value of x_i is 1; the right branch denotes the value 0.
- A special node c points to every clause gadget.
- For each clause, a clause gadget has four nodes. A node c_j points to three nodes for literals. Each literal node in turn points to a node in variable gadgets that makes the literal true.
- The designated start node b is the top node in the variable gadget for x_1. The bottom node of the variable gadget for x_k points to the special node c.

The game G starts by selecting values for variables $x_1, x_2, \ldots, x_k$. Player I selects values for $x_1, x_3, \ldots, x_{2h+1}, \ldots x_k$; Player II selects values for $x_2, x_4, \ldots, x_{2h}, \ldots, x_{k-1}$. Then Player II is forced to move to the special node c.
At the special node c, Player II tries to select a clause. If a clause is satisfied, all its literals are blocked by value nodes in variable gadgets. Player II will lose. If a clause is falsified, Player II can move to a value node in variable gadgets and win. Hence Player II tries to select a falsified clause. Hence ϕ is true if and only if Player I has a winning strategy in G. □

TM's with Sublinear Space

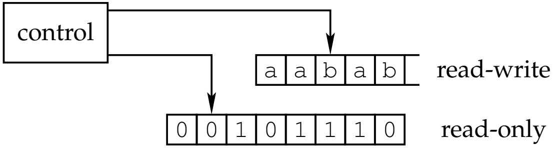


Figure: Schematics for TM's using Sublinear Space

- For sublinear space, we consider TM's with two tapes.
  - a read-only input tape containing the input string; and
  - a read-write work tape.
- The input head cannot move outside the portion of the tape containing the input.
- The cells scanned on the work tape contribute to the space complexity.

Space Complexity Classes L and NL

Definition 13

$\underline{L}$ ($= SPACE(\log n)$) is the class of languages decidable by a TM in logarithmic space.
$\underline{NL}$ ($= NSPACE(\log n)$) is the class of languages decidable by an NTM in logarithmic space.

Example 14

$A = \{\mathtt{0}^k\mathtt{1}^k : k \geq 0\} \in L$.

Proof.

Consider
$M =$ "On input w:

1. Check if w is of the form $\mathtt{0}^*\mathtt{1}^*$. If not, reject.
2. Count the number of $\mathtt{0}$'s and $\mathtt{1}$'s on the work tape.
3. If they are equal, accept; otherwise, reject." □

PATH is in *NL*

Example 15

Recall $PATH = \{\langle G, s, t\rangle : G$ is a directed graph with a path from s to $t\}$. Show $PATH \in NL$.

Proof.

Consider
N = "On input $\langle G, s, t\rangle$ where G is a directed graph with nodes s and t:

1. Repeat m times (m is the number of nodes in G)
   1. Nondeterministically select the next node for the path. If the next node is t, accept.
2. Reject.

N only needs to store the current node on the work tape. Hence N runs in space $O(\lg n)$. □

- We do not know if $PATH \in L$.

Configurations of TM's with Sublinear Space

Definition 16

Let M be a TM with a separate read-only input tape and w an input string. A configuration of M on w consists of a state, the contents of work tape, and locations of the two tape heads.

- Note that the input w is no longer a part of the configuration.
- If M runs in space $f(n)$ and $|w| = n$, the number of configurations of M on w is $n2^{O(f(n))}$.
  - Suppose M has q states and g tape symbols. The number of configurations is at most $qnf(n)g^{f(n)} \in n2^{O(f(n))}$.
- Note that when $f(n) \geq \lg n$, $n2^{O(f(n))} = 2^{O(f(n))}$.

Savitch's Theorem Revisited

- Recall that we assume $f(n) \geq n$ in the theorem.
- We can in fact relax the assumption to $f(n) \geq \lg n$.
- The proof is identical except that we are simulating an NTM N with a read-only input tape.
- When $f(n) \geq \lg n$, the depth of recursion is $\lg(n2^{O(f(n))}) = \lg n + O(f(n)) = O(f(n))$. At each level, $\lg(n2^{O(f(n))}) = O(f(n))$ space is needed.
- Hence $NSPACE(f(n)) \subseteq SPACE(f^2(n))$ when $f(n) \geq \lg n$.

Log Space Reducibility

Definition 17

A <u>log space transducer</u> is a TM with a read-only input tape, a write-only output tape, and a read-write work tape. The work tape may contain $O(\lg n)$ symbols.

Definition 18

$f : \Sigma^* \rightarrow \Sigma^*$is a <u>log space computable function</u> if there is a log space transducer that halts with $f(w)$ in its work tape on every input w.

Definition 19

A language A is <u>log space reducible</u> to a language B (written $A \leq_L B$) if there is a log space computable function f such that $w \in A$ if and only if $f(w) \in B$ for every w.

Properties about Log Space Reducibility

Theorem 20

If $A \leq_L B$ and $B \in L$, $A \in L$.

Proof.

Let a TM M_B decide B in space $O(\lg n)$. Consider

$M_A =$ "On input w:

1. Compute the first symbol of $f(w)$.
2. Simulate M_B on the current symbol.
3. If M_B ever changes its input head, compute the symbol of $f(w)$ at the new location.
    - More precisely, restart the computation of $f(w)$ and ignore all symbols of $f(w)$ except the one needed by M_B.
4. If M_B accepts, accepts; otherwise, reject.

□

- Can we write down $f(w)$ on M_B's work tape?
    - No. $f(w)$ may need more than logarithmic space.

Properties about Log Space Reducibility

Theorem 20

If $A \leq_L B$ and $B \in L$, $A \in L$.

Proof.

Let a TM M_B decide B in space $O(\lg n)$. Consider
$M_A =$ "On input w:

1. Compute the first symbol of $f(w)$.
2. Simulate M_B on the current symbol.
3. If M_B ever changes its input head, compute the symbol of $f(w)$ at the new location.
   - More precisely, restart the computation of $f(w)$ and ignore all symbols of $f(w)$ except the one needed by M_B.
4. If M_B accepts, accepts; otherwise, reject.

□

- Can we write down $f(w)$ on M_B's work tape?
  - No. $f(w)$ may need more than logarithmic space.

NL-Completeness

Definition 21

A language B is NL-complete if

- $B \in NL$; and
- $A \leq_L B$ for every $A \in NL$.

- Note that we require $A \leq_L B$ instead of $A \leq_P B$.
- We will show $NL \subseteq P$ (Corollary 24).
- Hence every two problems in NL (except $\emptyset$ and Σ^*) are polynomial time reducible to each other (why?).

Corollary 22

If any NL-complete language is in L, then $L = NL$.

NL-Completeness

Theorem 23

PATH is NL-complete.

Proof.

Let an NTM M decide A in $O(\lg n)$ space. We assume M has a unique accepting configuration. Given w, we construct $\langle G, s, t\rangle$ in log space such that M accepts w if and only if G has a path from s to t.

Nodes of G are configurations of M on w. For configurations c_1 and c_2, the edge (c_1, c_2) is in G if c_1 yields c_2 in M. s and t are the start and accepting configurations of M on w respectively.

Clearly, M accepts w if and only if G has a path from s to t. It remains to show that G can be computed by a log space transducer. Observe that a configuration of M on w can be represented in $c \lg n$ space for some c. The transducer simply enumerates all string of legnth $c \lg n$ and outputs those that are configurations of M on w. The edges (c_1, c_2)'s are computed similarly. The transducer only needs to read the tape contents under the head locations in c_1 to decide whether c_1 yields c_2 in M. □

$NL \subseteq P$

Corollary 24

$NL \subseteq P$.

Proof.

A TM using space $f(n)$ has at most $n2^{O(f(n))}$ configurations and hence runs in time $n2^{O(f(n))}$. A log space transducer therefore runs in polynomial time. Hence any problem in NL is polynomial time reducible to $PATH$. The result follows by $PATH \in P$. $\square$

- The polynomial time reduction in the proof of Theorem 9 can be computed in log space.
- Hence $TQBF$ is $PSPACE$-complete with respect to log space reducibility.

$NL = coNL$

Theorem 25 (Immerman–Szelepcsényi)

$NL = coNL$.

Proof.

We will give an NTM M deciding $\overline{PATH}$ in space $O(\lg n)$. Hence $\overline{PATH} \in NL$. Recall that $PATH$ is NL-complete. For any $A \in NL$, we have $A \leq_L PATH$. Hence $\overline{A} \leq_L \overline{PATH}$. Since $\overline{PATH} \in NL$, $\overline{A} \in NL$. That is, $\overline{\overline{A}} = A \in coNL$. We have $NL \subseteq coNL$. For any $B \in coNL$, we have $\overline{B} \in NL$. Hence $\overline{B} \leq_L PATH$. Thus $B = \overline{\overline{B}} \leq_L \overline{PATH}$. Since $\overline{PATH} \in NL$, we have $B \in NL$. We have $coNL \subseteq NL$.

$NL = coNL$

Proof (cont'd).

```
Input: On ⟨G, s, t⟩
c_0 = 1;
// G has m nodes
foreach i = 0, ..., m − 1 do
    c_{i+1} = 1 ;            // c_{i+1} counts the nodes reached from s in ≤ i + 1 steps
    foreach node v ≠ s in G do
        d = 0 ;              // d recounts the nodes reached from s in ≤ i steps
        foreach node u in G do
            Nondeterministically continue;
            Nondeterministically follow a path of length ≤ i from s;
            Reject if the path does not end at u;
            d = d + 1;
            if (u, v) is an edge in G then
                c_{i+1} = c_{i+1} + 1;
                break;
        end
        if d ≠ c_i then Reject;
        ;                    // check if the result is correct
    end
end
```

$NL = coNL$

Proof (cont'd).

$d = 0$; // d recounts the nodes reached from s

foreach node u in G **do**

Nondeterministically **continue**;

Nondeterministically follow a path of length $\leq m$ from s;

Reject if the path does not end at u;

if $u = t$ **then** Reject;

; // do not count t

$d = d + 1$;

end

if $d \neq c_m$ **then** Reject;

else Accept;

;

The NTM M counts the nodes reached from s in the first phrase. The variable c_i is the number of nodes reached from s in $\leq i$ steps. Initially, $c_0 = 1$. To compute c_{i+1} from c_i, M goes through each node $v \neq s$ in G. For each v, M tries to find all nodes reached from s in $\leq i$ steps. For each such node u, M increments d. It also increments c_{i+1} if u points to v. If $d = c_i$, M has found all node reached from s in $\leq i$ steps. Hence c_{i+1} is correct. M proceeds to compute c_{i+2}.

At the second phrase, M counts nodes reached from s but excluding t. If s reaches the

L, NL, P, and $PSPACE$

- The relationship between different complexity classes now becomes

$$L \subseteq NL = coNL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$$

- We will prove $NL \subsetneq PSPACE$ in the next chapter.
- Hence at least on inclusion is propcer.
  - But we do not know which one.