بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه چهاردهم: پیچیدگی حافظه (۴)

Theory of computation - 002 - S14 - space complexity (4), NL=coNL

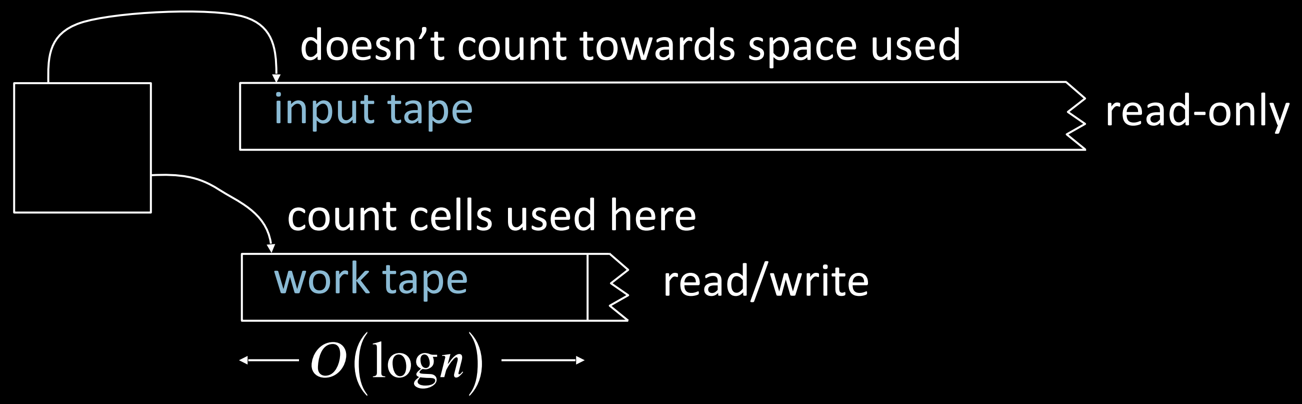
Review: log space

Model: 2-tape TM with read-only input tape for defining sublinear space computation.

Defn: L = SPACE$(\log n)$

NL = NSPACE$(\log n)$

Log space can represent a constant number of pointers into the input.

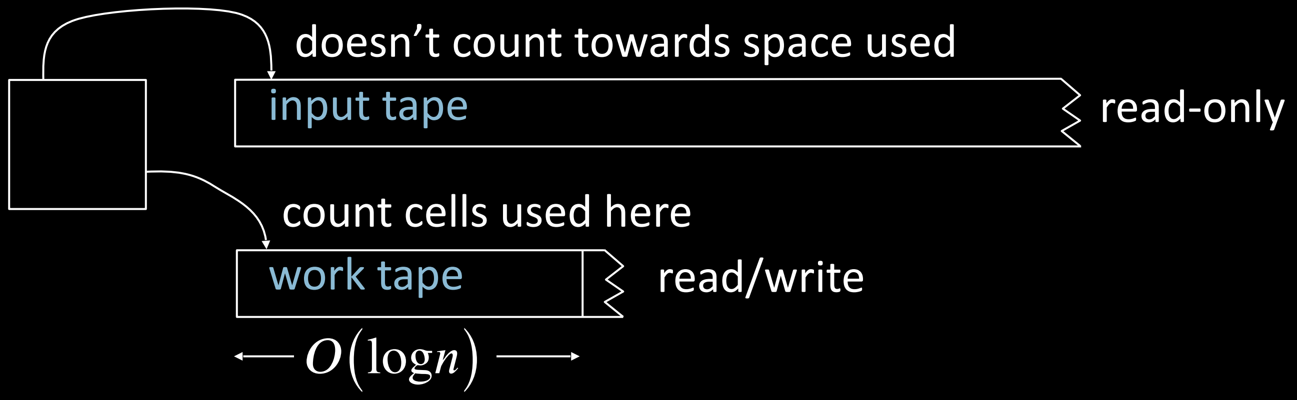


Review: log space

Model: 2-tape TM with read-only input tape for defining sublinear space computation.

Defn: L = SPACE$(\log n)$

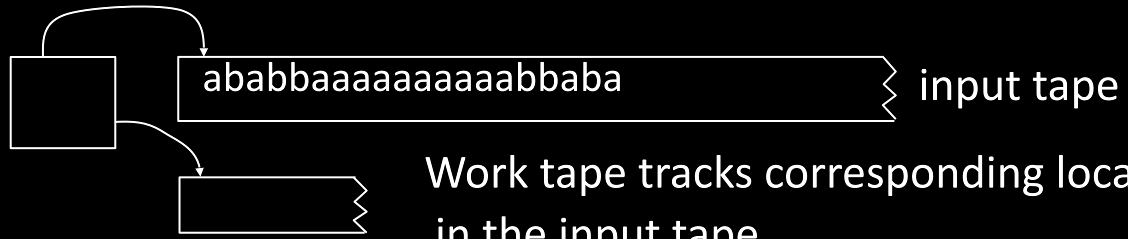
NL = NSPACE$(\log n)$



Log space can represent a constant number of pointers into the input.

Examples

1. $\{ww^{\mathcal{R}} \mid w \in \Sigma^*\} \in$ L



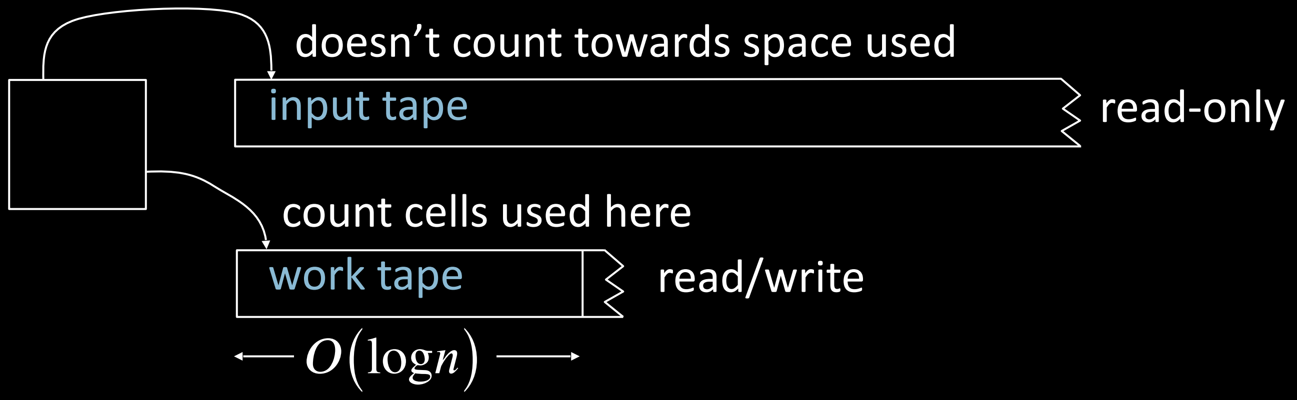
Work tape tracks corresponding locations in the input tape.

Review: log space

Model: 2-tape TM with read-only input tape for defining sublinear space computation.

Defn: L = SPACE$(\log n)$

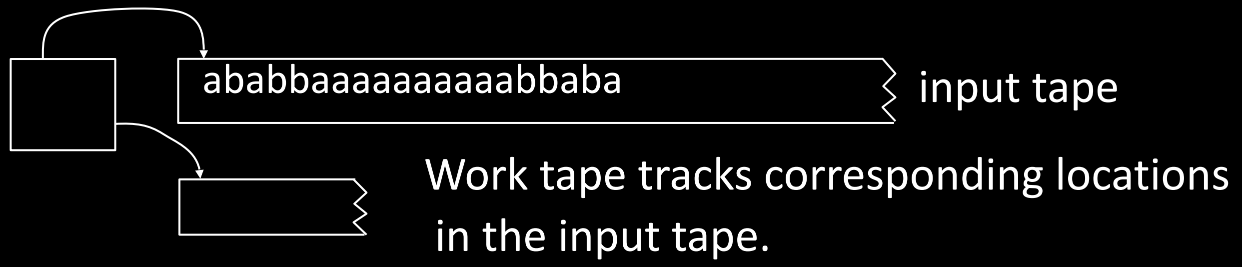
NL = NSPACE$(\log n)$



Log space can represent a constant number of pointers into the input.

Examples

1. $\{ww^{\mathcal{R}} \mid w \in \Sigma^*\} \in$ L



2. $PATH \in$ NL

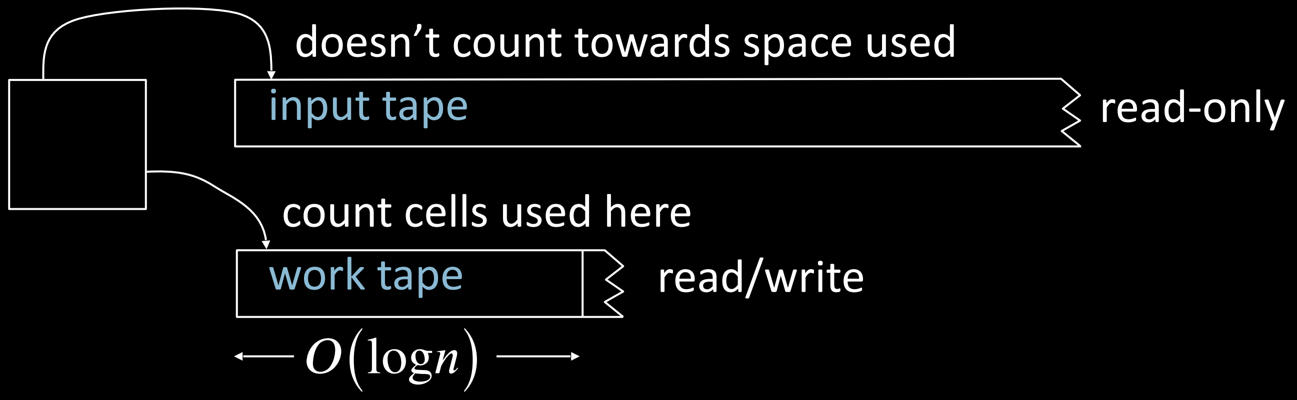
Nondeterministically select the nodes of a path connecting s to t.

Review: log space

Model: 2-tape TM with read-only input tape for defining sublinear space computation.

Defn: L = SPACE$(\log n)$

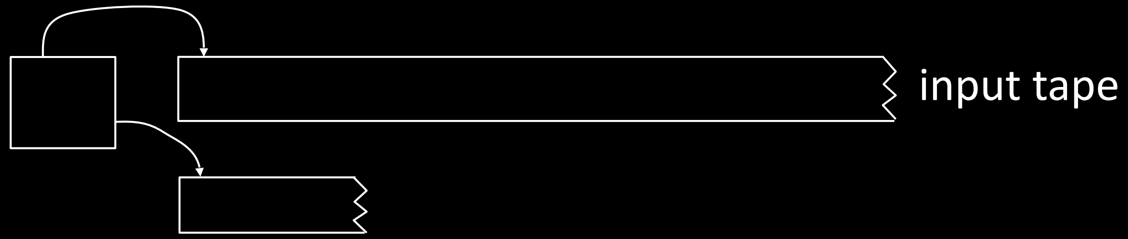
NL = NSPACE$(\log n)$



Log space can represent a constant number of pointers into the input.

Examples

1. $\{ww^{\mathcal{R}} \mid w \in \Sigma^*\} \in$ L



2. $PATH \in$ NL

Nondeterministically select the nodes of a path connecting s to t.

$$G = \big((v_3, v_5),\ (v_7,\ v_{22}),\ \ldots\big),\ s = \cdots,\ t = \cdots$$

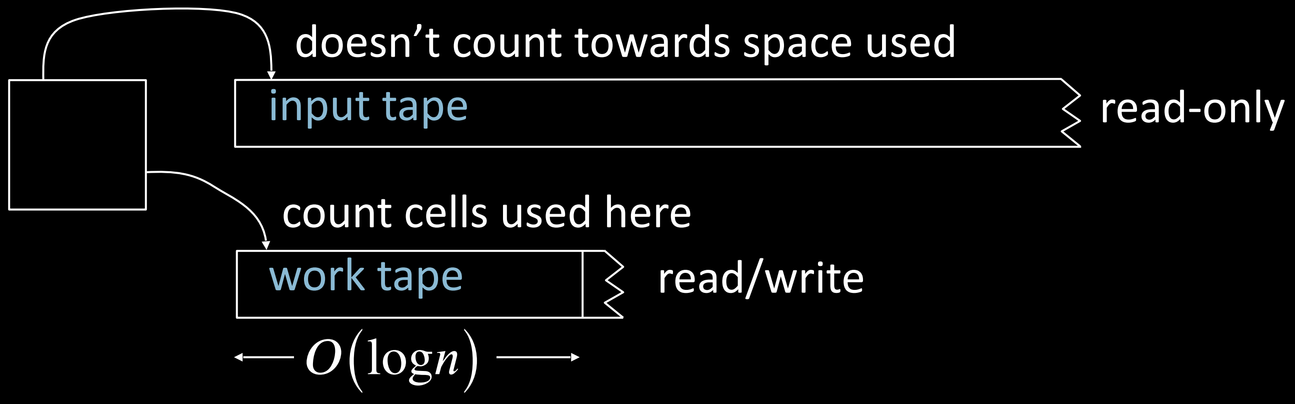
Work tape tracks the current node on the guessed path.

Review: log space

Model: 2-tape TM with read-only input tape for defining sublinear space computation.

Defn: L = SPACE$(\log n)$

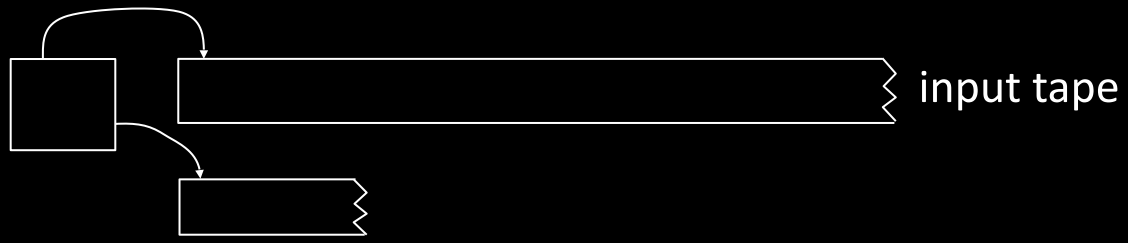
NL = NSPACE$(\log n)$



Log space can represent a constant number of pointers into the input.

Examples

1. $\{ww^{\mathscr{R}} \mid w \in \Sigma^*\} \in$ L

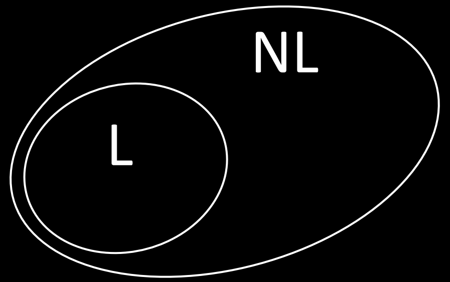


2. $PATH \in$ NL

Nondeterministically select the nodes of a path connecting s to t.

$G = \left((v_3, v_5),\ (v_7,\ v_{22}),\ \ldots \right),\ s = \cdots,\ t = \cdots$

Work tape tracks the current node on the guessed path.



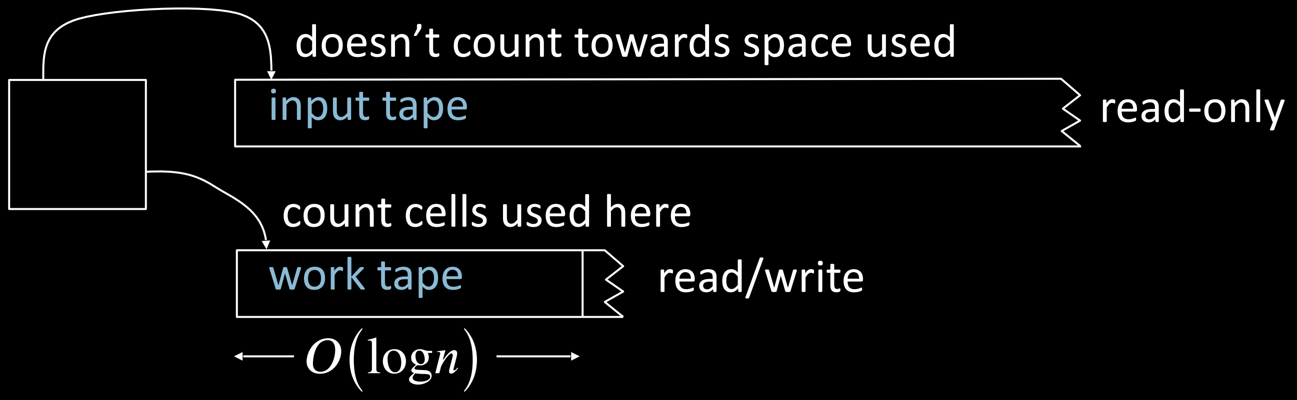
L = NL? Unsolved

Review: log space

Model: 2-tape TM with read-only input tape for defining sublinear space computation.

Defn: L = SPACE$(\log n)$

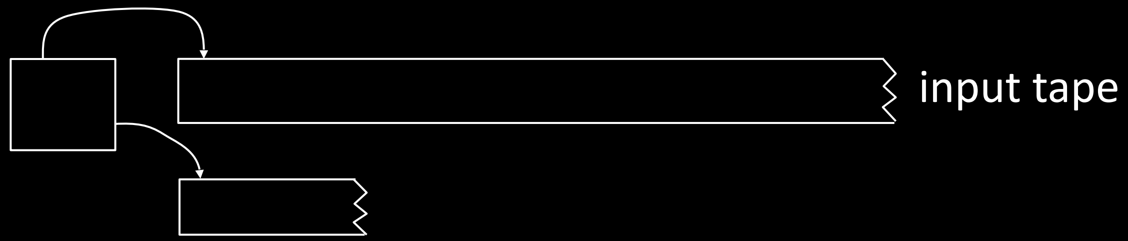
NL = NSPACE$(\log n)$



Log space can represent a constant number of pointers into the input.

Examples

1. $\{ww^{\mathscr{R}} \mid w \in \Sigma^*\} \in$ L

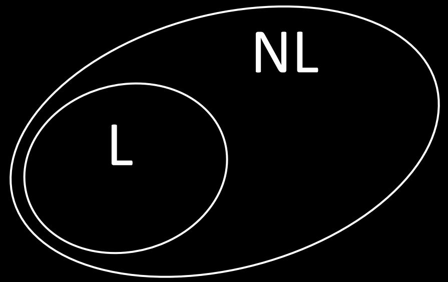


2. $PATH \in$ NL

Nondeterministically select the nodes of a path connecting s to t.

$$G = \big((v_3, v_5),\ (v_7,\ v_{22}),\ \ldots\big),\ s = \cdots,\ t = \cdots$$

Work tape tracks the current node on the guessed path.



L = NL? Unsolved

Review: L ⊆ P

Theorem: L ⊆ P

Review: L ⊆ P

Theorem: L ⊆ P

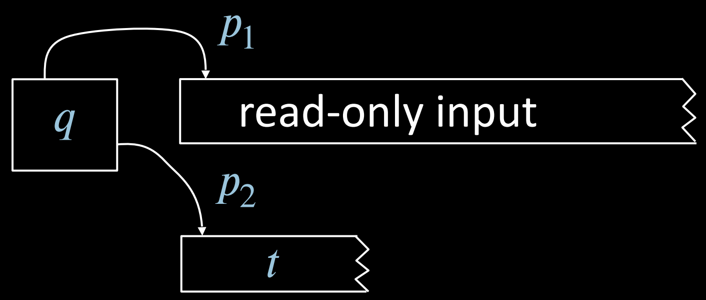
Proof: Say M decides A in space $O(\log n)$.

Review: L ⊆ P

Theorem: L ⊆ P

Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the work tape contents.



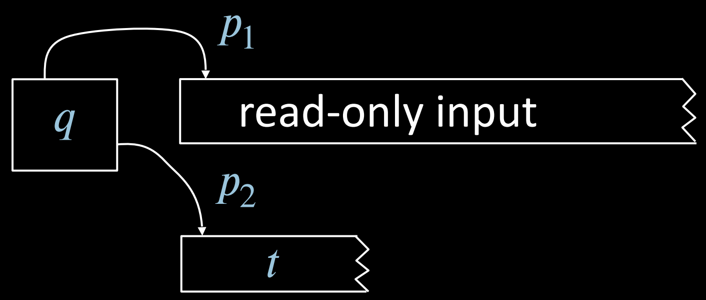
Review: L ⊆ P

Theorem: L ⊆ P

Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the work tape contents.

The number of such configurations is $|Q| \times n \times O(\log n) \times d^{O(\log n)} = O(n^k)$ for some k.



Review: L ⊆ P

Theorem: L ⊆ P

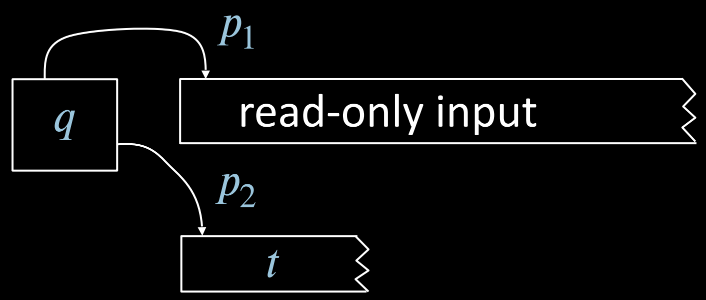
Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the work tape contents.

The number of such configurations is $|Q| \times n \times O(\log n) \times d^{O(\log n)} = O(n^k)$ for some k.

Therefore M runs in polynomial time.

Conclusion: $A \in$ P



Review: L ⊆ P

Theorem: L ⊆ P

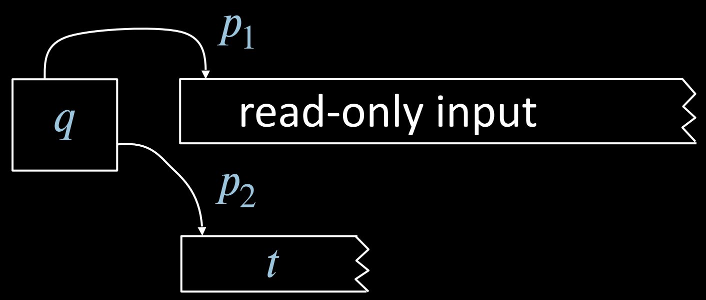
Proof: Say M decides A in space $O(\log n)$.

Defn: A configuration for M on w is (q, p_1, p_2, t) where q is a state, p_1 and p_2 are the tape head positions, and t is the work tape contents.

The number of such configurations is $|Q| \times n \times O(\log n) \times d^{O(\log n)} = O(n^k)$ for some k.

Therefore M runs in polynomial time.

Conclusion: $A \in$ P



Review: NL $\subseteq$ SPACE$\left(\log^2 n\right)$

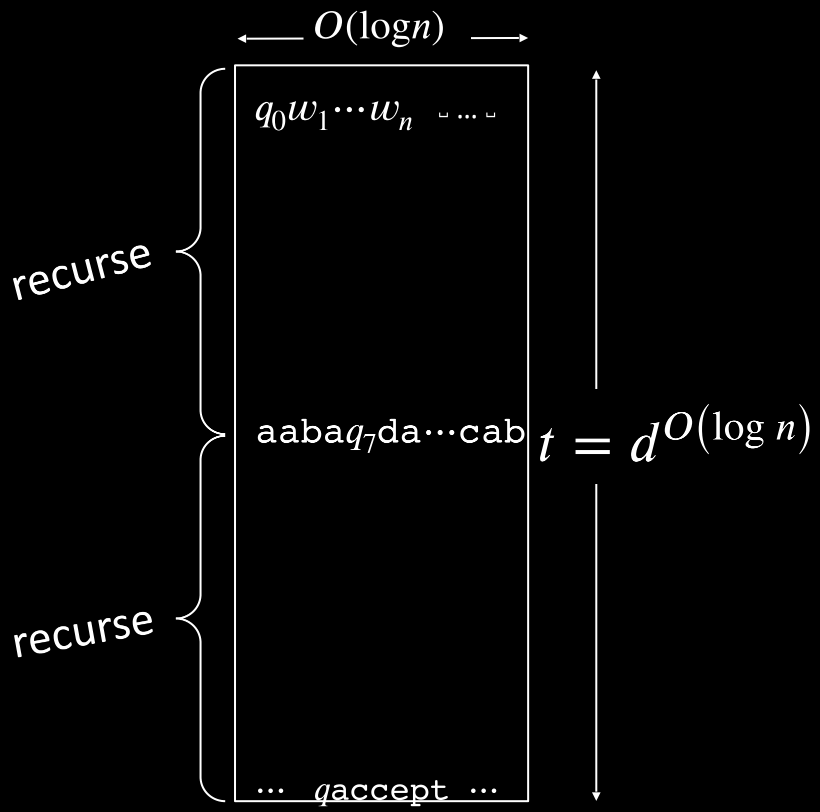
Theorem: NL $\subseteq$ SPACE$\left(\log^2 n\right)$

Proof: Savitch's theorem works for log space

Review: NL $\subseteq$ SPACE$\left(\log^2 n\right)$

Theorem: NL $\subseteq$ SPACE$\left(\log^2 n\right)$

Proof: Savitch's theorem works for log space



Review: NL $\subseteq$ SPACE$\left(\log^2 n\right)$

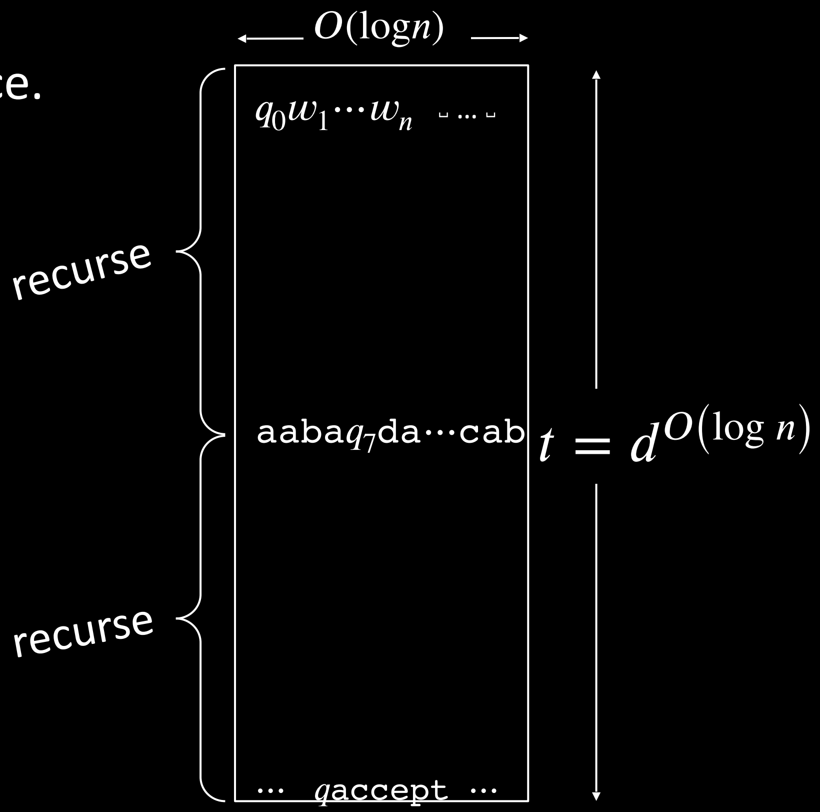
Theorem: NL $\subseteq$ SPACE$\left(\log^2 n\right)$

Proof: Savitch's theorem works for log space

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

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

Total $O\left(\log^2 n\right)$ space.



Review: NL $\subseteq$ SPACE$\left(\log^2 n\right)$

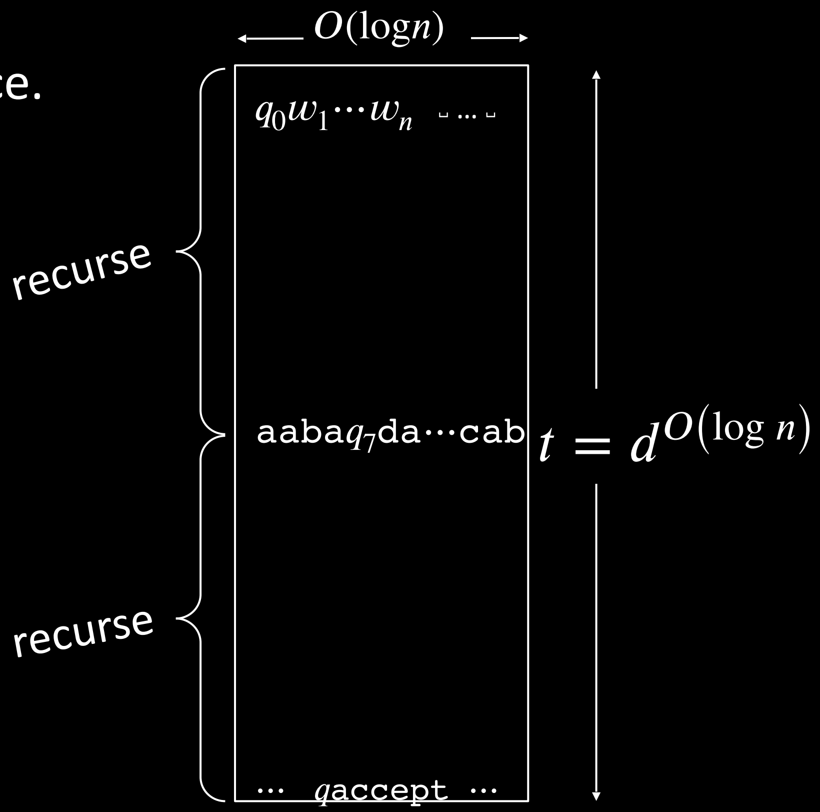
Theorem: NL $\subseteq$ SPACE$\left(\log^2 n\right)$

Proof: Savitch's theorem works for log space

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

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

Total $O\left(\log^2 n\right)$ space.



Review: NL $\subseteq$ P

Theorem: NL $\subseteq$ P

Review: NL ⊆ P

Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

Review: NL ⊆ P

Theorem: NL ⊆ P

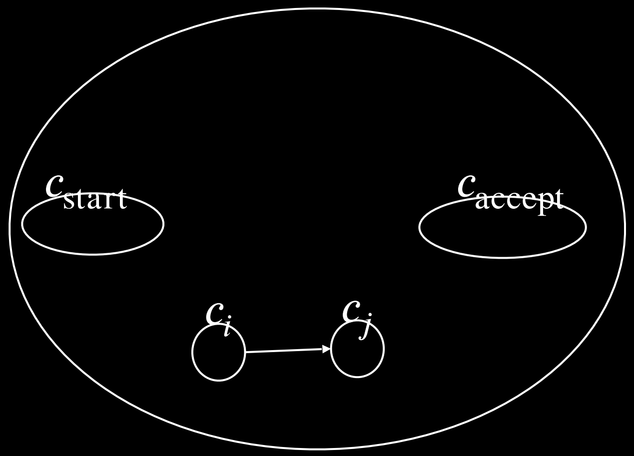
Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.

configuration graph $G_{M,w}$



Review: NL ⊆ P

Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

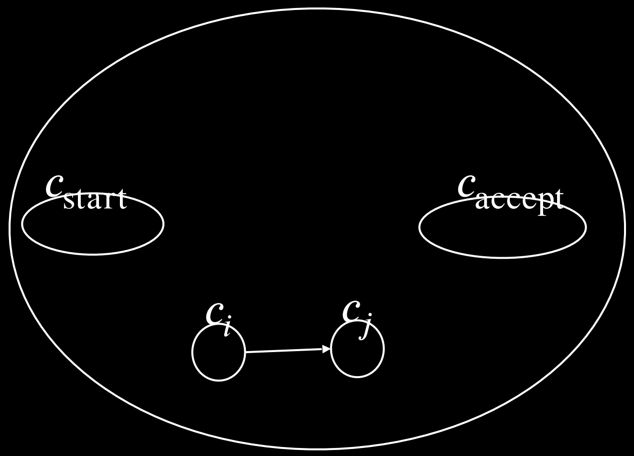
Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \rightarrow c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

configuration graph $G_{M,w}$



Review: NL ⊆ P

Theorem: NL ⊆ P

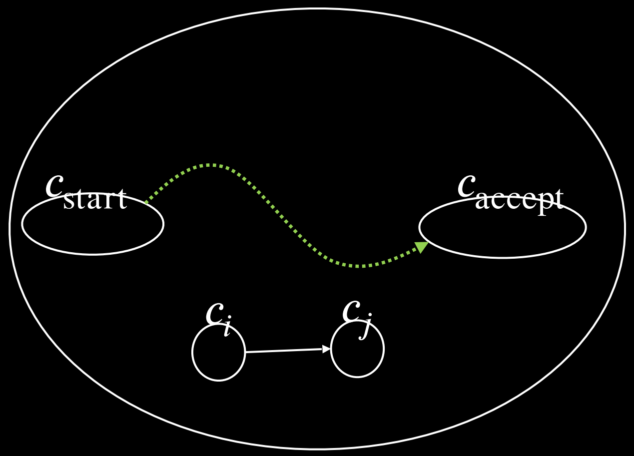
Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

- **nodes:** all configurations for M on w
- **edges:** edge from $c_i \rightarrow c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

configuration graph $G_{M,w}$



iff M accepts w

Review: NL ⊆ P

Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

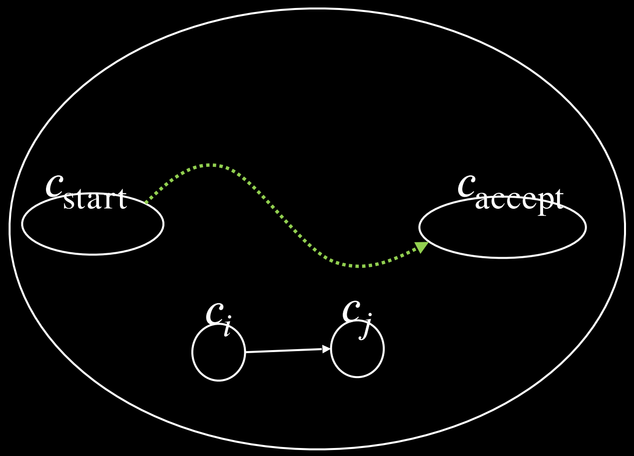
Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

- **nodes:** all configurations for M on w
- **edges:** edge from $c_i \rightarrow c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

Polynomial time algorithm T for A:

configuration graph $G_{M,w}$



iff M accepts w

Review: NL ⊆ P

Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

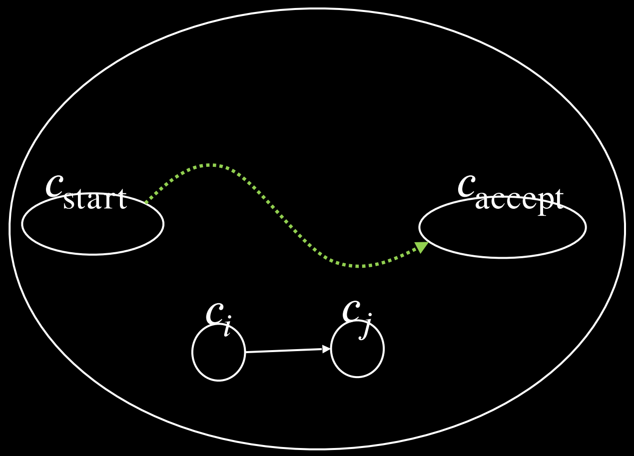
Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

configuration graph $G_{M,w}$



iff M accepts w

Polynomial time algorithm T for A:

$T =$ "On input w

1. Construct $G_{M,w}$. [polynomial size]
2. *Accept* if there is a path from c_{start} to c_{accept}.
   Reject if not."

Review: NL ⊆ P

Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The configuration graph $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.

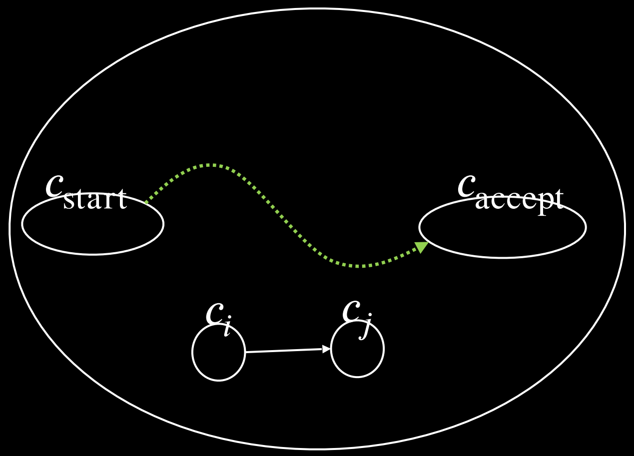
Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

Polynomial time algorithm T for A:

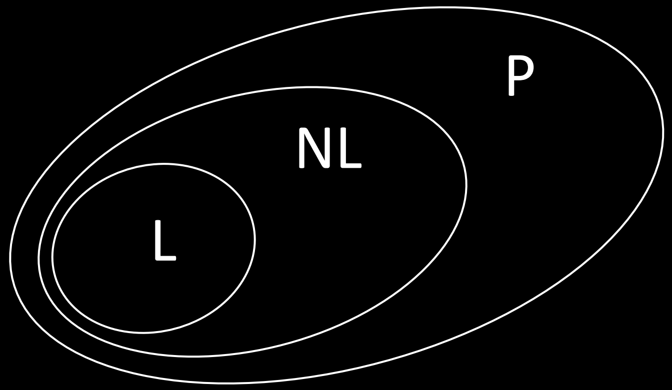
$T =$ "On input w

1. Construct $G_{M,w}$. [polynomial size]
2. *Accept* if there is a path from c_{start} to c_{accept}.
   Reject if not."

configuration graph $G_{M,w}$



iff M accepts w



Review: NL ⊆ P

Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

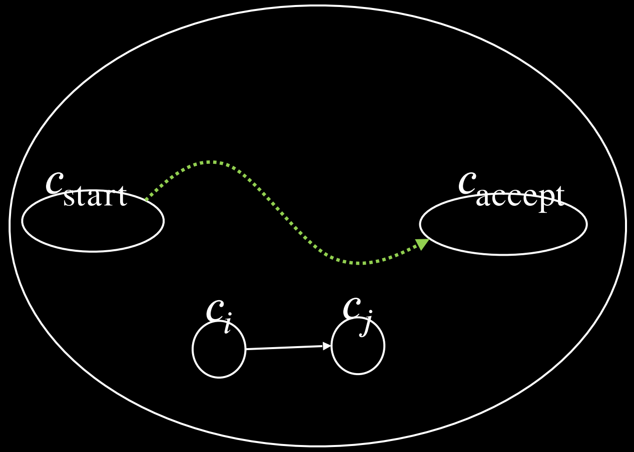
Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

configuration graph $G_{M,w}$

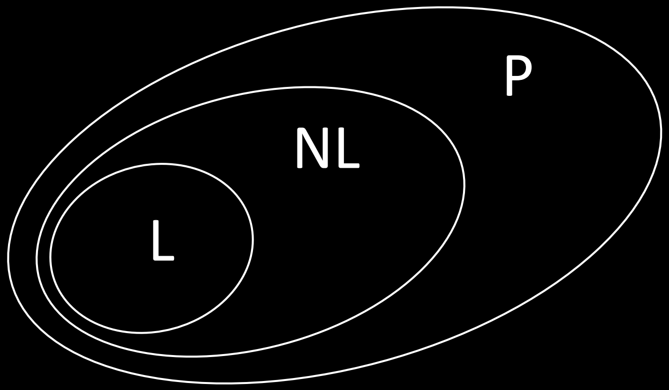


iff M accepts w

Polynomial time algorithm T for A:

$T =$ "On input w

1. Construct $G_{M,w}$. [polynomial size]
2. *Accept* if there is a path from c_{start} to c_{accept}.
   Reject if not."



L = P? Unsolved

Review: NL ⊆ P

Theorem: NL ⊆ P

Proof: Say NTM M decides A in space $O(\log n)$.

Defn: The <u>configuration graph</u> $G_{M,w}$ for M on w has

nodes: all configurations for M on w

edges: edge from $c_i \to c_j$ if c_i can yield c_j in 1 step.

Claim: M accepts w iff the configuration graph $G_{M,w}$ has a path from c_{start} to c_{accept}

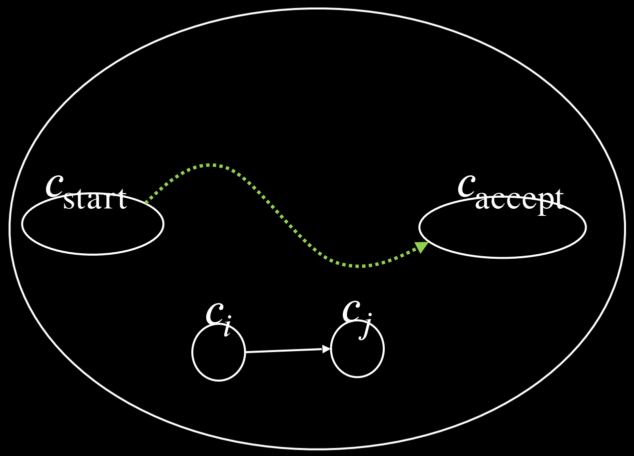
Polynomial time algorithm T for A:

$T =$ "On input w

1. Construct $G_{M,w}$. [polynomial size]
2. *Accept* if there is a path from c_{start} to c_{accept}.

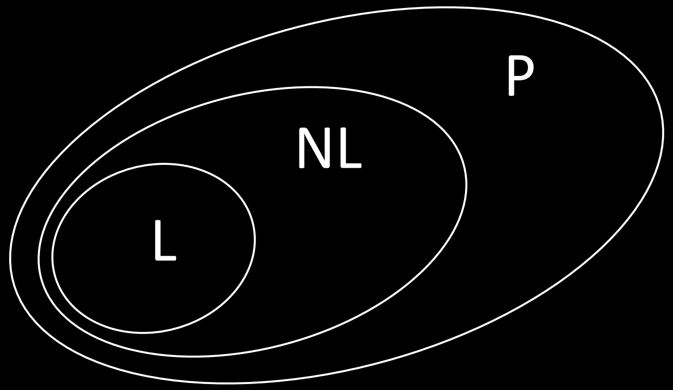
   Reject if not."

configuration graph $G_{M,w}$



iff M accepts w

L = P? Unsolved



NL-completeness

Defn: B is NL-complete if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\mathrm{L}} B$

NL-completeness

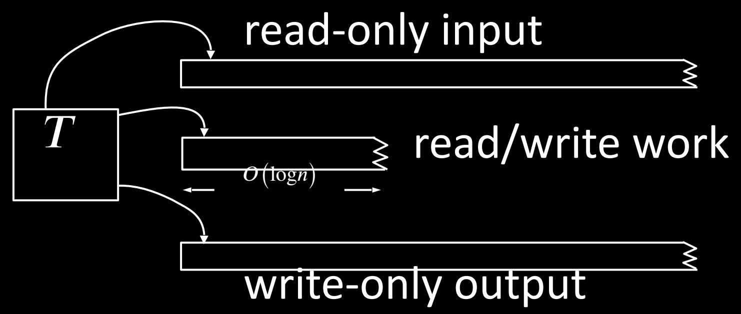
Defn: B is <u>NL-complete</u> if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\mathrm{L}} B$

Log-space reducibility

Defn: A <u>log-space transducer</u> is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape

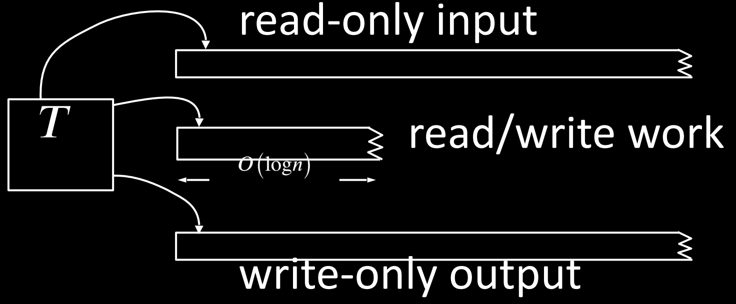


NL-completeness

Defn: B is NL-complete if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\mathrm{L}} B$

Log-space reducibility



Defn: A log-space transducer is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape

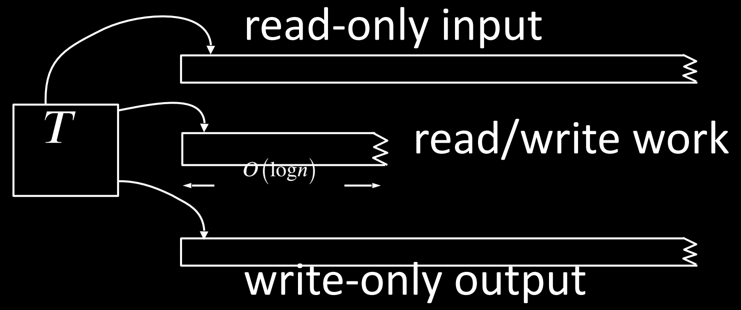
A log-space transducer T computes a function $f : \Sigma^* \rightarrow \Sigma^*$ if T on input w halts with $f(w)$ on its output tape for all w.
Say that f is computable in log-space.

NL-completeness

Defn: B is <u>NL-complete</u> if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\mathrm{L}} B$

Log-space reducibility



Defn: A <u>log-space transducer</u> is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape

A log-space transducer T computes a function $f : \Sigma^* \rightarrow \Sigma^*$ if T on input w halts with $f(w)$ on its output tape for all w.
Say that f is computable in log-space.

Defn: A is <u>log-space reducible</u> to B ($A \leq_{\mathrm{L}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in log-space.

NL-completeness

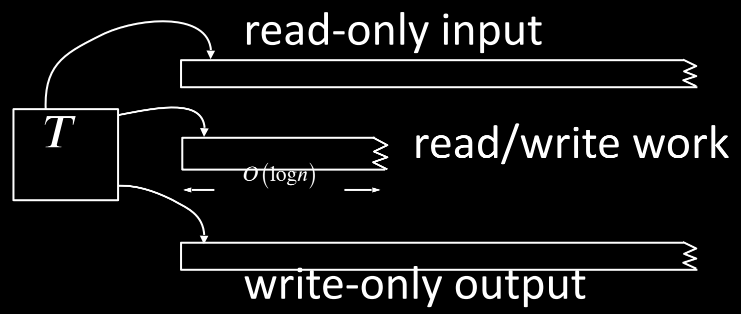
Defn: B is <u>NL-complete</u> if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\text{L}} B$

Log-space reducibility

Defn: A <u>log-space transducer</u> is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape



A log-space transducer T computes a function $f : \Sigma^* \to \Sigma^*$ if T on input w halts with $f(w)$ on its output tape for all w.
Say that f is computable in log-space.

Defn: A is <u>log-space reducible</u> to B ($A \leq_{\text{L}} B$) if $A \leq_{\text{m}} B$ by a reduction function that is computable in log-space.

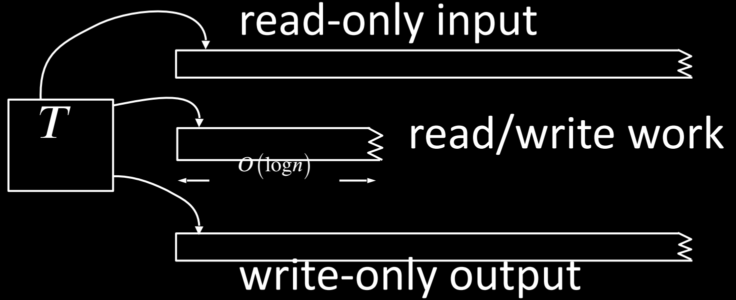
Theorem: If $A \leq_{\text{L}} B$ and $B \in$ L then $A \in$ L

NL-completeness

Defn: B is NL-complete if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_L B$

Log-space reducibility



Defn: A log-space transducer is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape

A log-space transducer T computes a function $f: \Sigma^* \to \Sigma^*$ if T on input w halts with $f(w)$ on its output tape for all w.
Say that f is computable in log-space.

Defn: A is log-space reducible to B ($A \leq_L B$) if $A \leq_m B$ by a reduction function that is computable in log-space.

Theorem: If $A \leq_L B$ and $B \in$ L then $A \in$ L

Proof: TM for $A =$ "On input w

NL-completeness

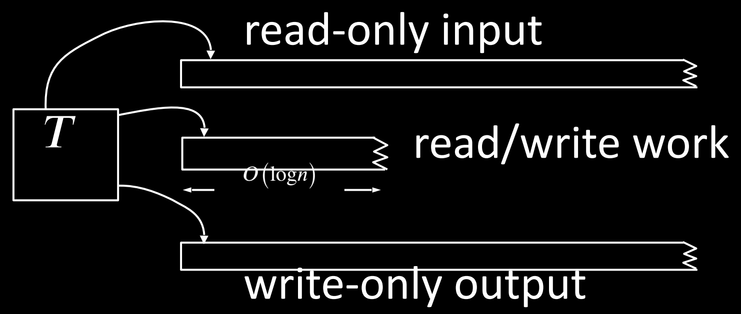
Defn: B is <u>NL-complete</u> if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\mathrm{L}} B$

Log-space reducibility

Defn: A <u>log-space transducer</u> is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape



A log-space transducer T computes a function $f: \Sigma^* \to \Sigma^*$ if T on input w halts with $f(w)$ on its output tape for all w.
Say that f is computable in log-space.

Defn: A is <u>log-space reducible</u> to B ($A \leq_{\mathrm{L}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in log-space.

Theorem: If $A \leq_{\mathrm{L}} B$ and $B \in$ L then $A \in$ L

Proof: TM for $A =$ "On input w

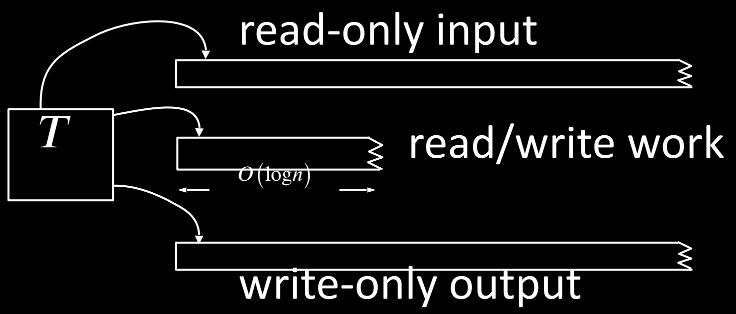
1. Compute $f(w)$

NL-completeness

Defn: B is <u>NL-complete</u> if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\mathrm{L}} B$

Log-space reducibility



Defn: A <u>log-space transducer</u> is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape

A log-space transducer T computes a function $f: \Sigma^* \to \Sigma^*$ if T on input w halts with $f(w)$ on its output tape for all w.
Say that f is computable in log-space.

Defn: A is <u>log-space reducible</u> to B ($A \leq_{\mathrm{L}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in log-space.

Theorem: If $A \leq_{\mathrm{L}} B$ and $B \in$ L then $A \in$ L

Proof: TM for $A =$ "On input w

1. Compute $f(w)$
2. Run decider for B on $f(w)$. Output same."

NL-completeness

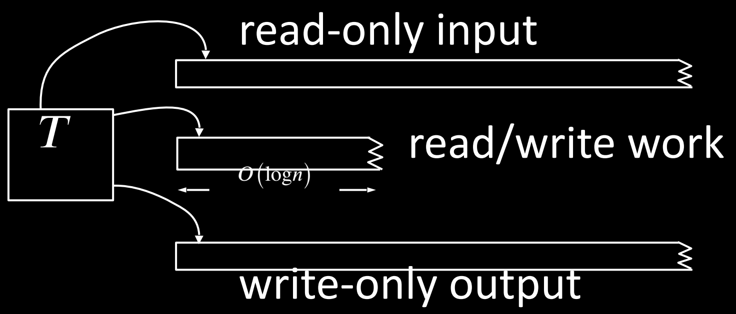
Defn: B is <u>NL-complete</u> if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\mathrm{L}} B$

Log-space reducibility

Defn: A <u>log-space transducer</u> is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape



A log-space transducer T computes a function $f: \Sigma^* \rightarrow \Sigma^*$ if T on input w halts with $f(w)$ on its output tape for all w.
Say that f is computable in log-space.

Defn: A is <u>log-space reducible</u> to B ($A \leq_{\mathrm{L}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in log-space.

Theorem: If $A \leq_{\mathrm{L}} B$ and $B \in$ L then $A \in$ L

Proof: TM for $A =$ "On input w

1. Compute $f(w)$
2. Run decider for B on $f(w)$. Output same."

BUT we don't have space to store $f(w)$.

NL-completeness

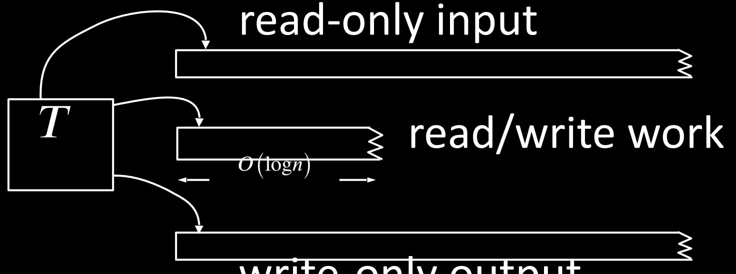
Defn: B is NL-complete if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\mathrm{L}} B$

Log-space reducibility

Defn: A log-space transducer is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape



A log-space transducer T computes a function $f: \Sigma^* \to \Sigma^*$ if T on input w halts with $f(w)$ on its output tape for all w.
Say that f is computable in log-space.

Defn: A is log-space reducible to B ($A \leq_{\mathrm{L}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in log-space.

Theorem: If $A \leq_{\mathrm{L}} B$ and $B \in$ L then $A \in$ L

Proof: TM for $A =$ "On input w

1. Compute $f(w)$
2. Run decider for B on $f(w)$. Output same."

BUT we don't have space to store $f(w)$.
So, (re-)compute symbols of $f(w)$ as needed.

NL-completeness

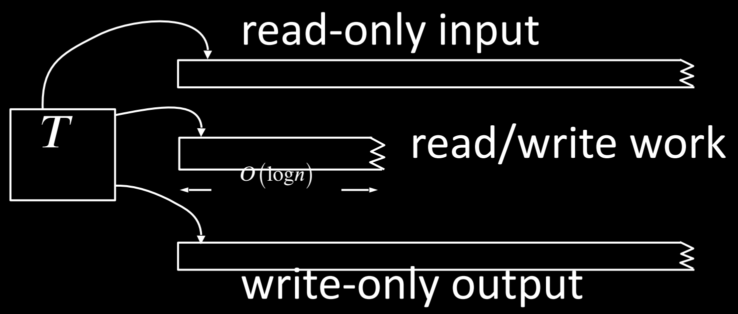
Defn: B is <u>NL-complete</u> if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\mathrm{L}} B$

Log-space reducibility

Defn: A <u>log-space transducer</u> is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape



A log-space transducer T computes a function $f: \Sigma^* \rightarrow \Sigma^*$ if T on input w halts with $f(w)$ on its output tape for all w.
Say that f is computable in log-space.

Defn: A is <u>log-space reducible</u> to B ($A \leq_{\mathrm{L}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in log-space.

Theorem: If $A \leq_{\mathrm{L}} B$ and $B \in$ L then $A \in$ L

Proof: TM for $A =$ "On input w

1. Compute $f(w)$
2. Run decider for B on $f(w)$. Output same."

BUT we don't have space to store $f(w)$.
So, (re-)compute symbols of $f(w)$ as needed.

Check-in 20.1

NL-completeness

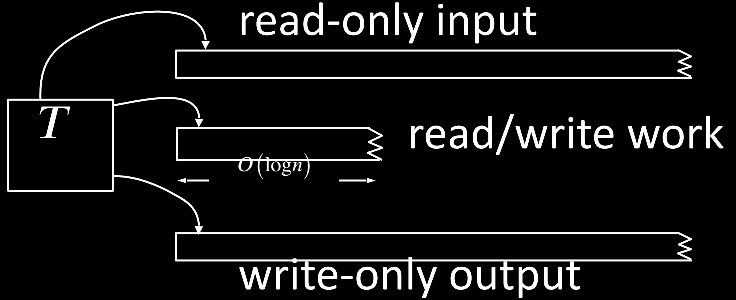
Defn: B is <u>NL-complete</u> if

1) $B \in$ NL
2) For all $A \in$ NL, $A \leq_{\mathrm{L}} B$

Log-space reducibility

Defn: A <u>log-space transducer</u> is a TM with three tapes:

1. read-only input tape of size n
2. read/write work tape of size $O(\log n)$
3. write-only output tape



Check-in 20.1

If T is a log-space transducer that computes f, then for inputs w of length n, how long can $f(w)$ be?

(a) at most $O(\log n)$
(b) at most $O(n)$
(c) at most polynomial in n
(d) at most $2^{O(n)}$
(e) any length

Σ^*

ll w.

m B

.

Theorem: If $A \leq_{\mathrm{L}} B$ and $B \in$ L then $A \in$ L

Proof: TM for $A =$ "On input w

1. Compute $f(w)$
2. Run decider for B on $f(w)$. Output same."

BUT we don't have space to store $f(w)$.

So, (re-)compute symbols of $f(w)$ as needed.

Check-in 20.1

$PATH$ is NL-complete

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

Give a log-space reduction f mapping A to $PATH$.

$$f(w) = \langle G, s, t \rangle$$

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

Give a log-space reduction f mapping A to $PATH$.

$$f(w) = \langle G, s, t \rangle$$

$w \in A$ iff G has a path from s to t

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_L PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

Give a log-space reduction f mapping A to $PATH$.

$$f(w) = \langle G, s, t \rangle = \langle G_{M,w}, c_{\text{start}}, c_{\text{accept}} \rangle$$

$w \in A$ iff G has a path from s to t

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

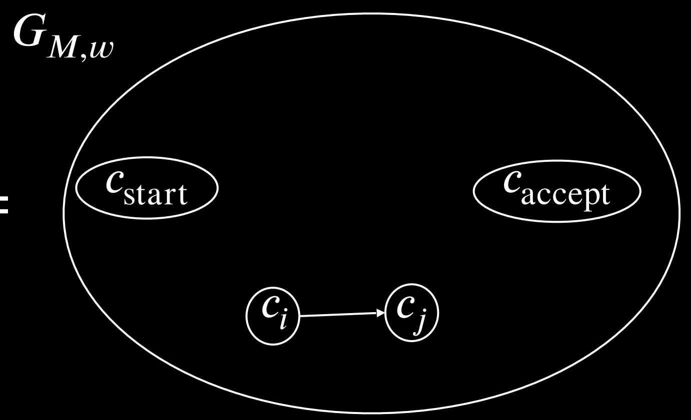
Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

Give a log-space reduction f mapping A to $PATH$.

$$f(w) = \langle G, s, t\rangle = \langle G_{M,w}, c_{\text{start}}, c_{\text{accept}}\rangle$$

$w \in A$ iff G has a path from s to t

$f(w) =$ 

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

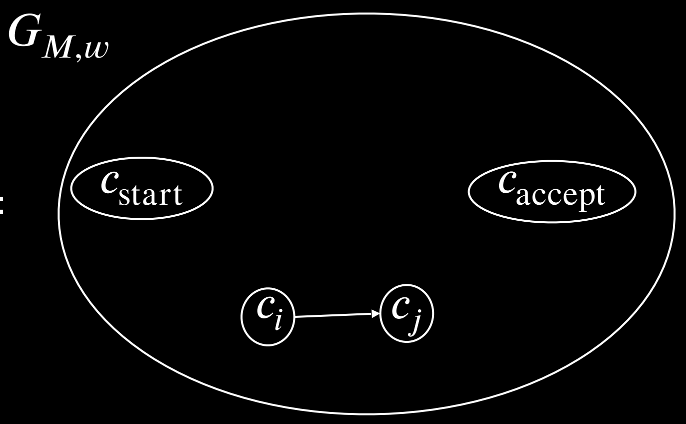
Give a log-space reduction f mapping A to $PATH$.

$$f(w) = \langle G, s, t\rangle = \langle G_{M,w}, c_{\text{start}}, c_{\text{accept}}\rangle$$

$w \in A$ iff G has a path from s to t

Here is a log-space transducer T to compute f in log-space.

$f(w) =$



$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

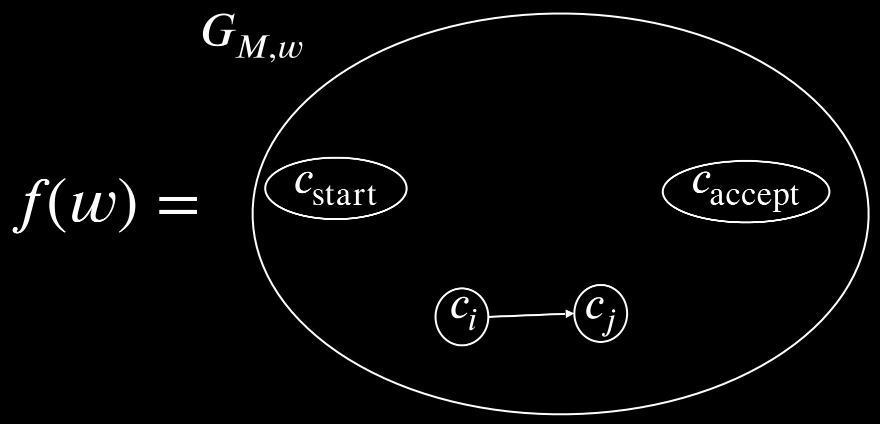
[Modify M to erase work tape and move heads to left end upon accepting.]

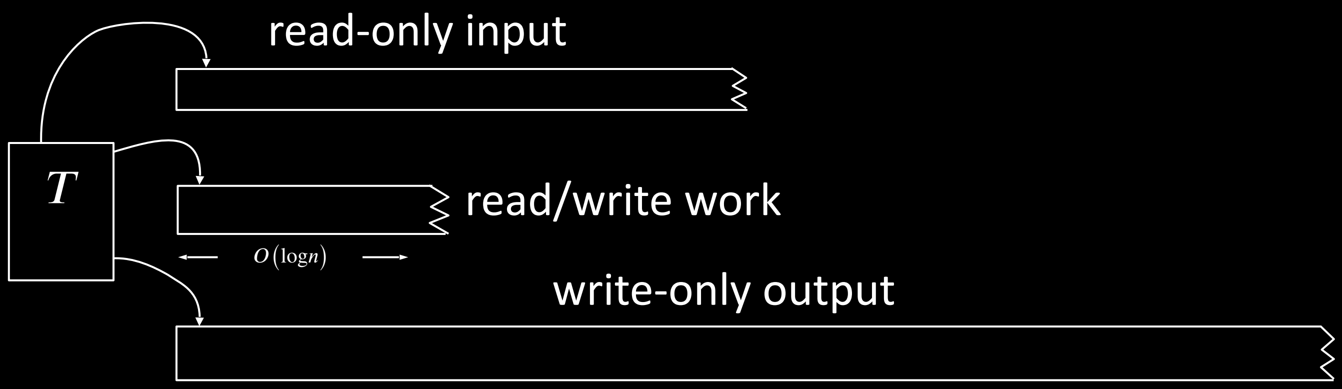
Give a log-space reduction f mapping A to $PATH$.

$$f(w) = \langle G, s, t \rangle = \langle G_{M,w}, c_{\text{start}}, c_{\text{accept}} \rangle$$

$w \in A$ iff G has a path from s to t

Here is a log-space transducer T to compute f in log-space.





$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\text{L}} PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

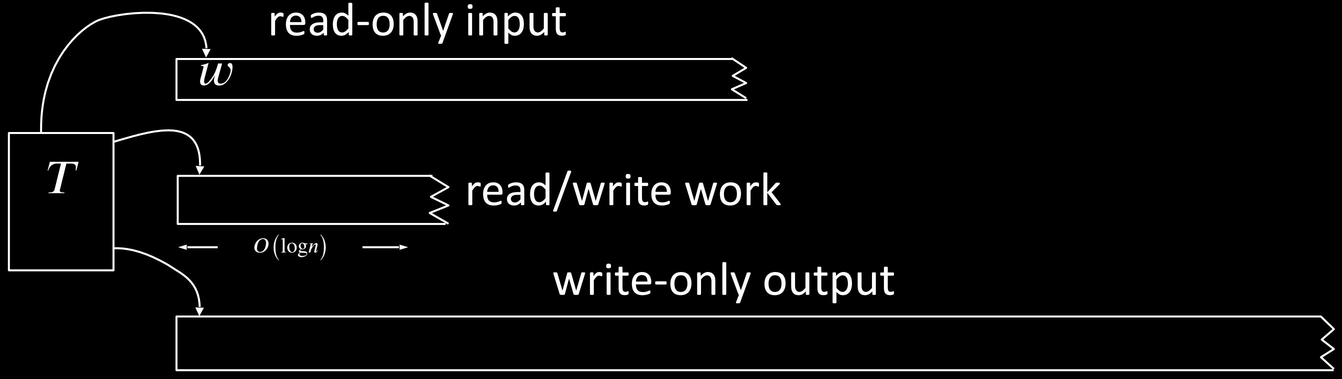
[Modify M to erase work tape and move heads to left end upon accepting.]

Give a log-space reduction f mapping A to $PATH$.

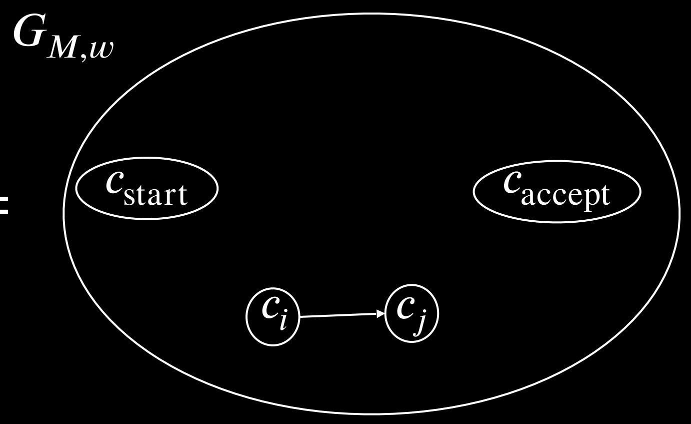
$$f(w) = \langle G, s, t \rangle = \langle G_{M,w}, c_{\text{start}}, c_{\text{accept}} \rangle$$

$w \in A$ iff G has a path from s to t

Here is a log-space transducer T to compute f in log-space.



$f(w) =$



$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_L PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

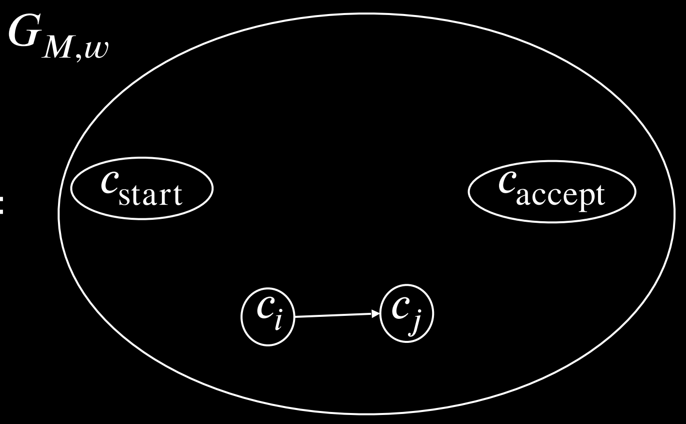
[Modify M to erase work tape and move heads to left end upon accepting.]

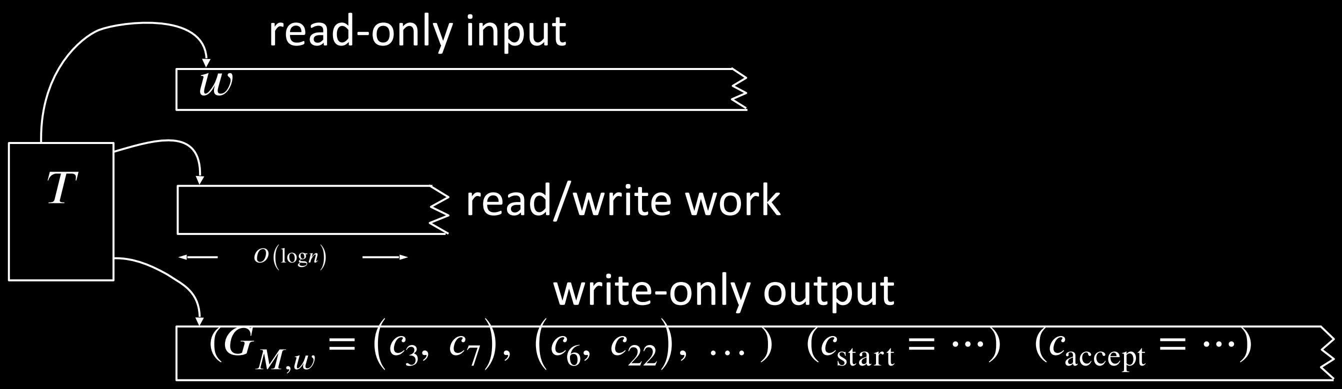
Give a log-space reduction f mapping A to $PATH$.

$$f(w) = \langle G, s, t \rangle = \langle G_{M,w}, c_{\text{start}}, c_{\text{accept}} \rangle$$

$w \in A$ iff G has a path from s to t

Here is a log-space transducer T to compute f in log-space.

$f(w) =$ 



$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

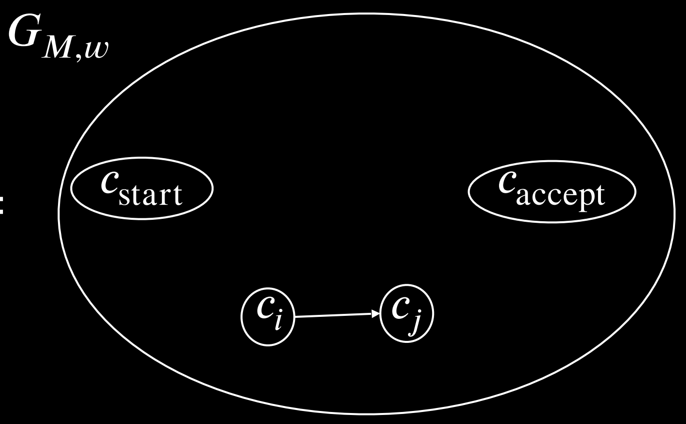
Give a log-space reduction f mapping A to $PATH$.

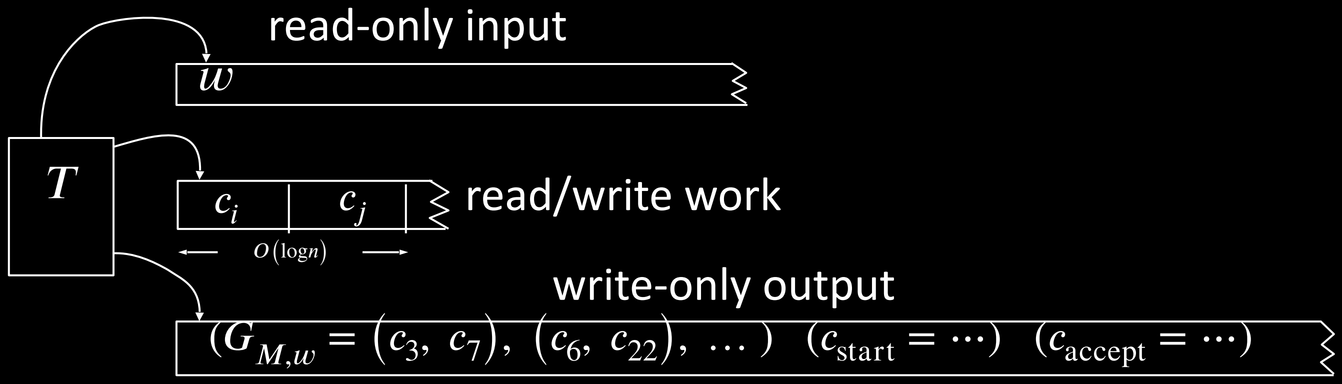
$$f(w) = \langle G, s, t \rangle = \langle G_{M,w}, c_{\text{start}}, c_{\text{accept}} \rangle$$

$w \in A$ iff G has a path from s to t

Here is a log-space transducer T to compute f in log-space.

$f(w) =$





$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_L PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

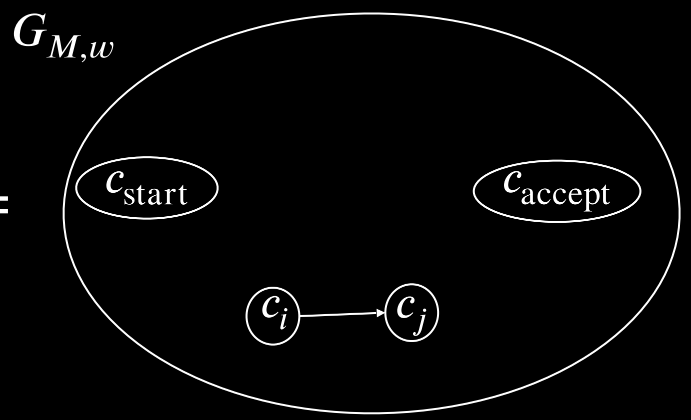
Give a log-space reduction f mapping A to $PATH$.

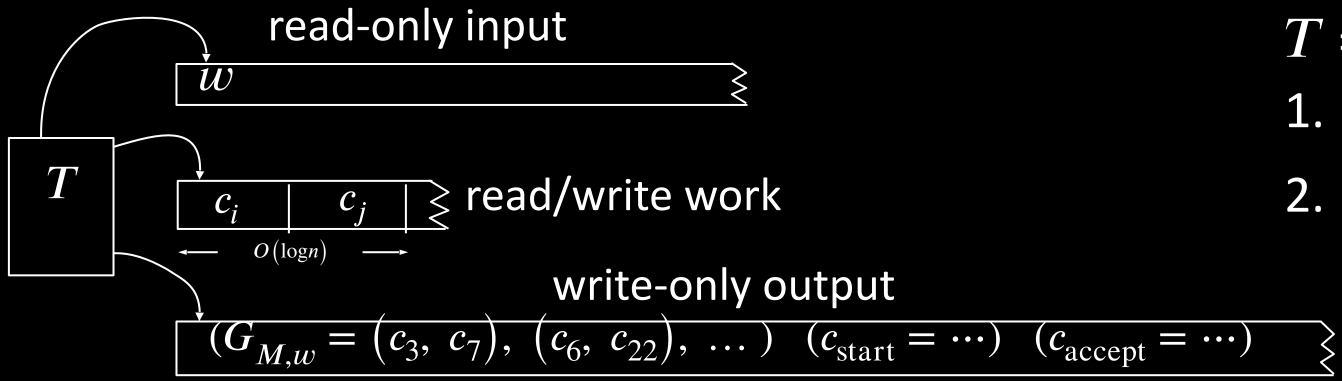
$$f(w) = \langle G, s, t \rangle = \langle G_{M,w}, c_{\text{start}}, c_{\text{accept}} \rangle$$

$w \in A$ iff G has a path from s to t

Here is a log-space transducer T to compute f in log-space.

$f(w) =$





$T =$ "on input w

1. For all pairs c_i, c_j of configurations of M on w.
2. Output those pairs which are legal moves for M.

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_L PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

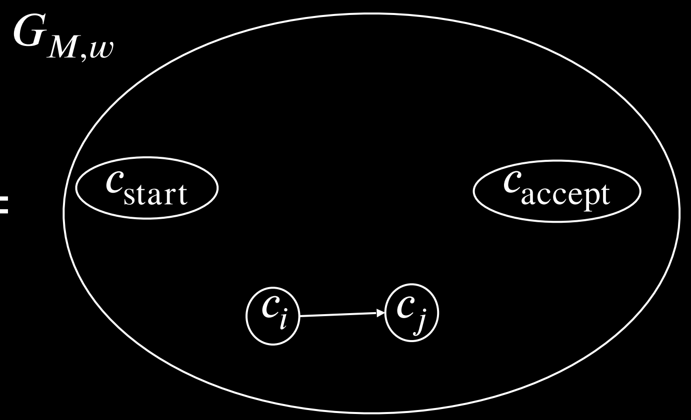
Give a log-space reduction f mapping A to $PATH$.

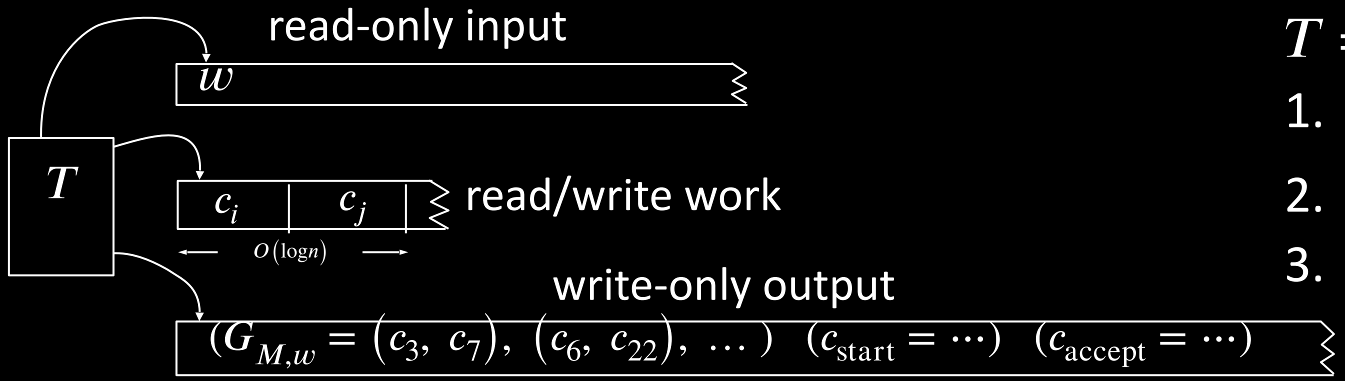
$$f(w) = \langle G, s, t \rangle = \langle G_{M,w}, c_{\text{start}}, c_{\text{accept}} \rangle$$

$w \in A$ iff G has a path from s to t

Here is a log-space transducer T to compute f in log-space.

$f(w) =$





$T =$ "on input w

1. For all pairs c_i, c_j of configurations of M on w.
2. Output those pairs which are legal moves for M.
3. Output c_{start} and c_{accept}."

$PATH$ is NL-complete

Theorem: $PATH$ is NL-complete

Proof: 1) $PATH \in$ NL ✓

2) For all $A \in$ NL, $A \leq_{\mathrm{L}} PATH$

Let $A \in$ NL be decided by NTM M in space $O(\log n)$.

[Modify M to erase work tape and move heads to left end upon accepting.]

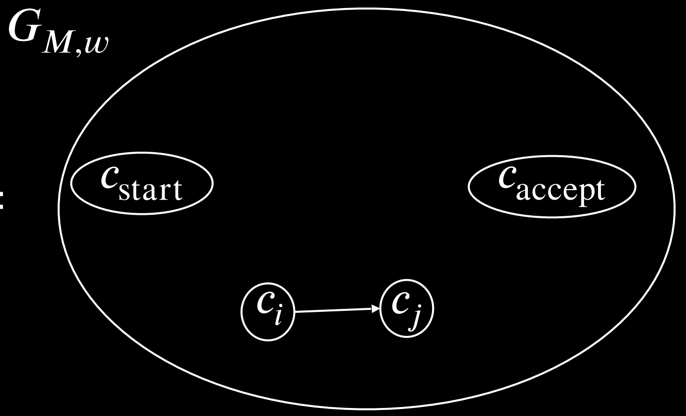
Give a log-space reduction f mapping A to $PATH$.

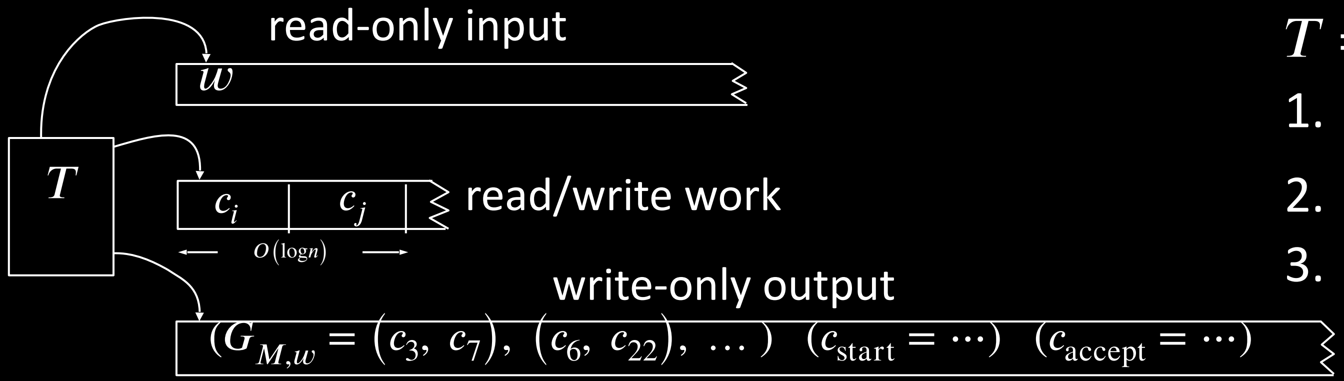
$$f(w) = \langle G, s, t \rangle = \langle G_{M,w}, c_{\text{start}}, c_{\text{accept}} \rangle$$

$w \in A$ iff G has a path from s to t

Here is a log-space transducer T to compute f in log-space.

$f(w) =$





$T =$ "on input w

1. For all pairs c_i, c_j of configurations of M on w.
2. Output those pairs which are legal moves for M.
3. Output c_{start} and c_{accept}."

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f : \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f: \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

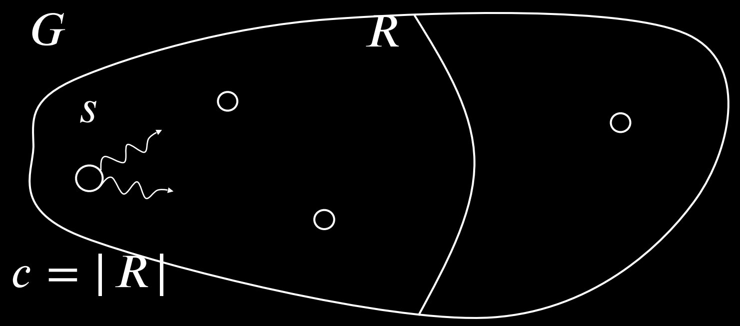
Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f: \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

Let $path(G, s, t) = \begin{cases} \text{YES, if } G \text{ has a path from } s \text{ to } t \\ \text{NO, if not} \end{cases}$

NL = coNL (part 1/4)



Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f: \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

Let $path(G, s, t) = \{$ YES, if G has a path from s to t
NO, if not

Let $R = R(G, s) = \{u \mid path(G, s, u) =$ YES$\}$

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

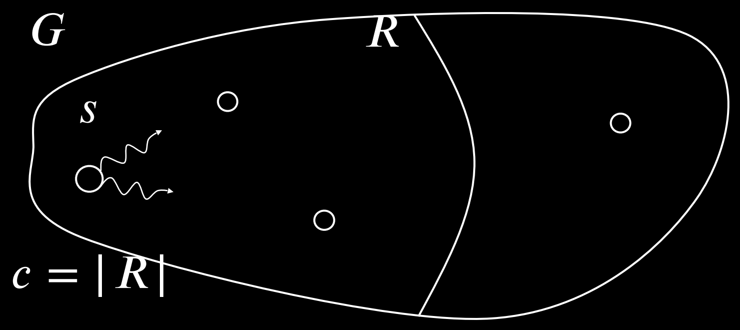
Defn: NTM M computes function $f\colon \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

Let $path(G, s, t) = \{$ YES, if G has a path from s to t
NO, if not

Let $R = R(G, s) = \{u \mid path(G, s, u) =$ YES$\}$

Let $c = c(G, s) = |R|$



R = Reachable nodes

c = # reachable

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f\colon \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

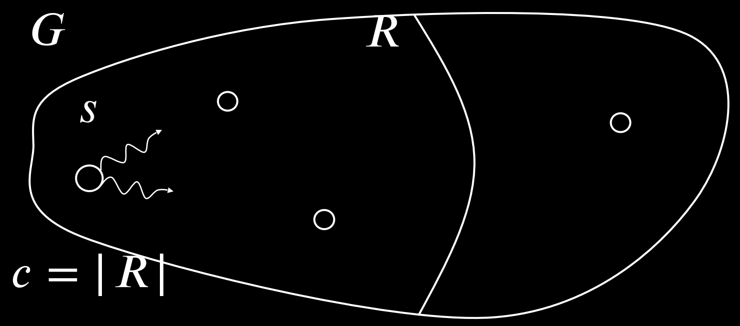
Let $path(G, s, t) = \begin{cases} \text{YES, if } G \text{ has a path from } s \text{ to } t \\ \text{NO, if not} \end{cases}$

Let $R = R(G, s) = \{u \mid path(G, s, u) = \text{YES}\}$

Let $c = c(G, s) = |R|$

R = Reachable nodes

c = # reachable



Theorem: If some NL-machine (log-space NTM) computes $path$, then some NL-machine computes c.

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f\colon \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

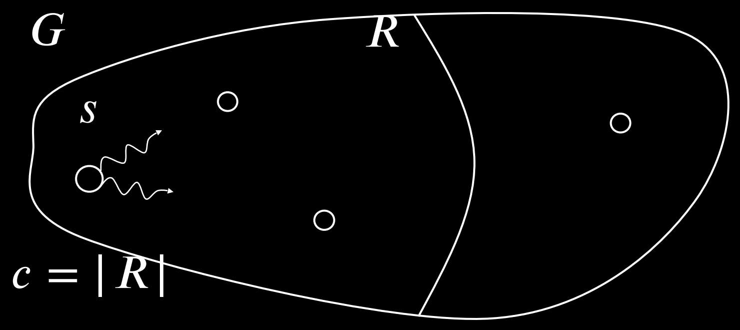
Let $path(G, s, t) = \begin{cases} \text{YES, if } G \text{ has a path from } s \text{ to } t \\ \text{NO, if not} \end{cases}$

Let $R = R(G, s) = \{u \mid path(G, s, u) = \text{ YES}\}$

Let $c = c(G, s) = |R|$

R = Reachable nodes

c = # reachable



Theorem: If some NL-machine (log-space NTM) computes $path$, then some NL-machine computes c.

Proof: “On input $\langle G, s\rangle$

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f : \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

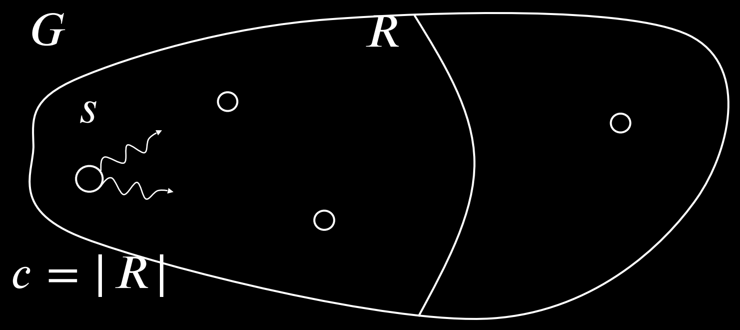
Let $path(G, s, t) = \begin{cases} \text{YES, if } G \text{ has a path from } s \text{ to } t \\ \text{NO, if not} \end{cases}$

Let $R = R(G, s) = \{u \mid path(G, s, u) = \text{YES}\}$

Let $c = c(G, s) = |R|$

R = Reachable nodes

c = # reachable



Theorem: If some NL-machine (log-space NTM) computes $path$, then some NL-machine computes c.

Proof: "On input $\langle G, s \rangle$

1. Let $k \leftarrow 0$

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f: \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

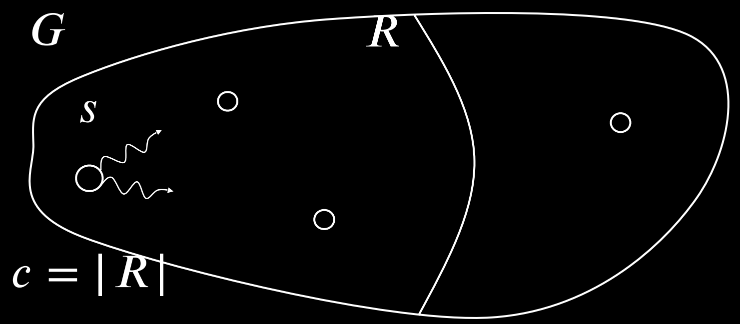
Let $path(G, s, t) = \begin{cases} \text{YES, if } G \text{ has a path from } s \text{ to } t \\ \text{NO, if not} \end{cases}$

Let $R = R(G, s) = \{u \mid path(G, s, u) = \text{YES}\}$

Let $c = c(G, s) = |R|$

R = Reachable nodes

c = # reachable



Theorem: If some NL-machine (log-space NTM) computes $path$, then some NL-machine computes c.

Proof: "On input $\langle G, s \rangle$

1. Let $k \leftarrow 0$
2. For each node u
3. If $path(G, s, u) =$ YES, then $k \leftarrow k + 1$
4. If $path(G, s, u) =$ NO, then continue

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f\colon \Sigma^* \rightarrow \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

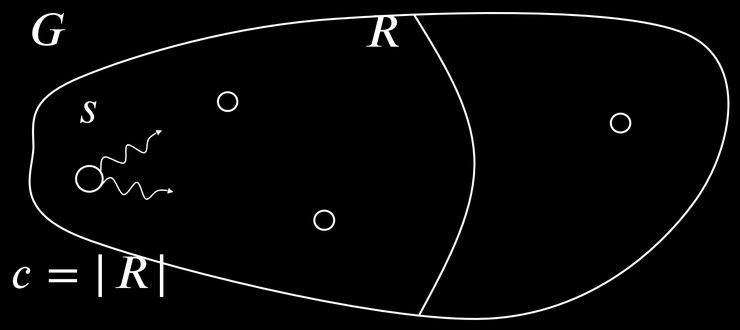
Let $path(G, s, t) = \begin{cases} \text{YES, if } G \text{ has a path from } s \text{ to } t \\ \text{NO, if not} \end{cases}$

Let $R = R(G, s) = \{u \mid path(G, s, u) = \text{YES}\}$

Let $c = c(G, s) = |R|$

R = Reachable nodes

c = # reachable



Theorem: If some NL-machine (log-space NTM) computes $path$, then some NL-machine computes c.

Proof: "On input $\langle G, s\rangle$

1. Let $k \leftarrow 0$
2. For each node u
3. If $path(G, s, u) =$ YES, then $k \leftarrow k + 1$
4. If $path(G, s, u) =$ NO, then continue
5. Output k"

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f: \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

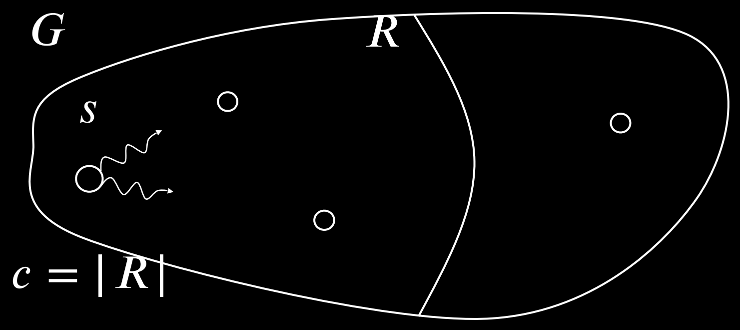
Let $path(G, s, t) = \{$ YES, if G has a path from s to t
NO, if not

Let $R = R(G, s) = \{u \mid path(G, s, u) =$ YES$\}$

Let $c = c(G, s) = |R|$

R = Reachable nodes

c = # reachable



Theorem: If some NL-machine (log-space NTM) computes $path$, then some NL-machine computes c.

Proof: "On input $\langle G, s \rangle$

1. Let $k \leftarrow 0$
2. For each node u
3. If $path(G, s, u) =$ YES, then $k \leftarrow k + 1$
4. If $path(G, s, u) =$ NO, then continue
5. Output k"

Next: Converse of above

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f\colon \Sigma^* \to \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

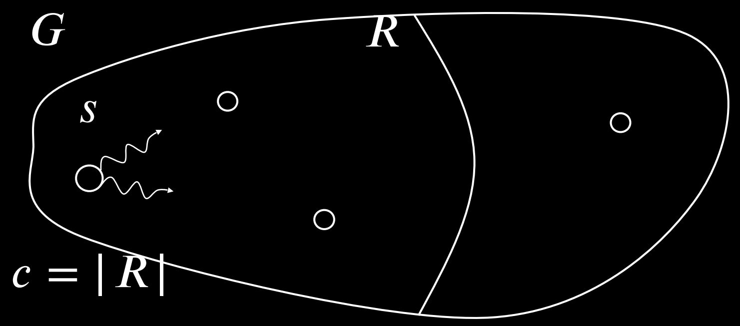
Let $path(G,s,t) = \begin{cases} \text{YES}, & \text{if } G \text{ has a path from } s \text{ to } t \\ \text{NO}, & \text{if not} \end{cases}$

Let $R = R(G,s) = \{u \mid path(G,s,u) = \text{YES}\}$

Let $c = c(G,s) = |R|$

R = Reachable nodes

c = # reachable



Theorem: If some NL-machine (log-space NTM) computes $path$, then some NL-machine computes c.

Proof: "On input $\langle G, s\rangle$

1. Let $k \leftarrow 0$
2. For each node u
3. If $path(G,s,u) =$ YES, then $k \leftarrow k+1$
4. If $path(G,s,u) =$ NO, then continue
5. Output k"

Next: Converse of above

Check-in 20.2

NL = coNL (part 1/4)

Theorem (Immerman-Szelepcsényi): NL = coNL

Proof: Show $\overline{PATH} \in$ NL

Defn: NTM M computes function $f : \Sigma^* \rightarrow \Sigma^*$ if for all w

1) All branches of M on w halt with $f(w)$ on the tape or reject.
2) Some branch of M on w does not reject.

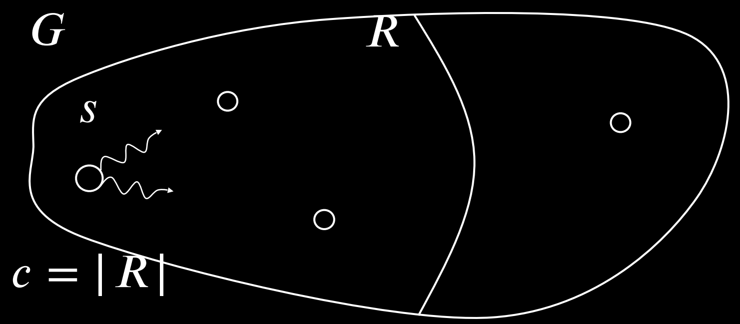
Let $path(G, s, t) = \begin{cases} \text{YES, if } G \text{ has a path from } s \text{ to } t \\ \text{NO, if not} \end{cases}$

Let $R = R(G, s) = \{u \mid path(G, s, u) = \text{YES}\}$

Let $c = c(G, s) = |R|$

R = Reachable nodes

c = # reachable



Check-in 20.2

Consider the statements:

(1) $\overline{PATH} \in$ NL, and

(2) Some NL-machine computes the $path$ function.

What implications can we prove *easily*?

(a) (1) $\rightarrow$ (2) only

(b) (2) $\rightarrow$ (1) only

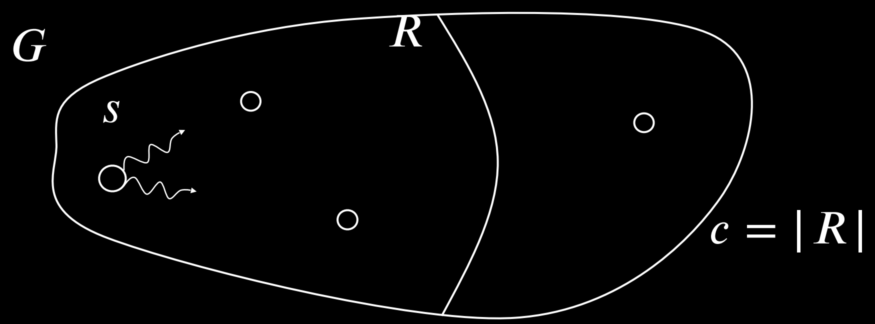
(c) Both implications

(d) Neither implication

Check-in 20.2

NL = coNL (part 2/4) – key idea

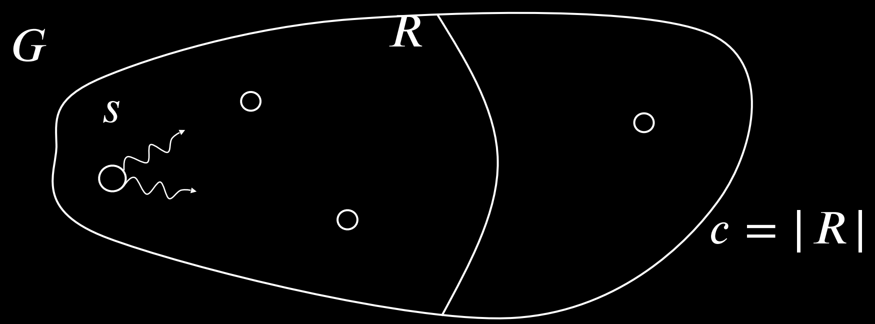
Theorem: If some NL-machine computes c, then some NL-machine computes $path$. c ===> path



NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$. c ===> path

Proof: "On input $\langle G, s, t\rangle$

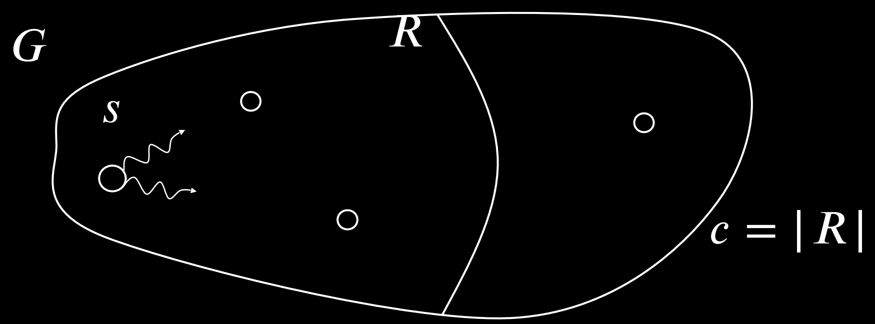


NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$. c ===> path

Proof: "On input $\langle G, s, t\rangle$

1. Compute c



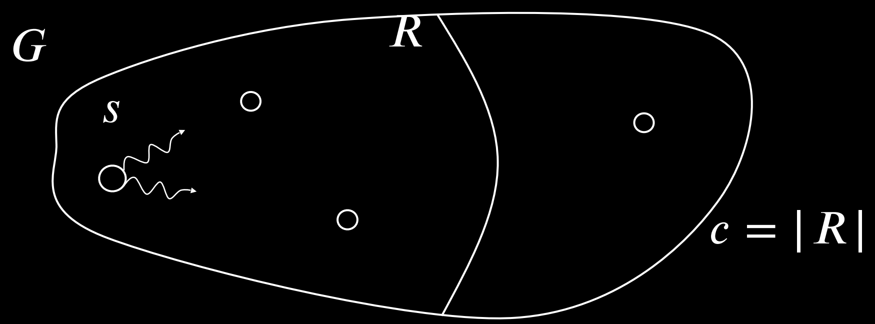
NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$.

c ===> path

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$

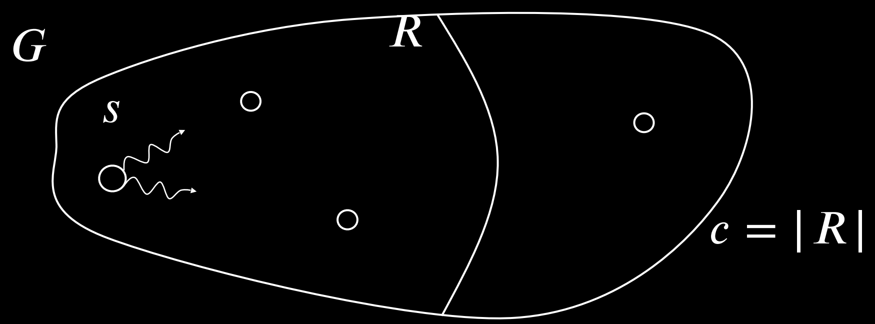


NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$. c ===> path

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u



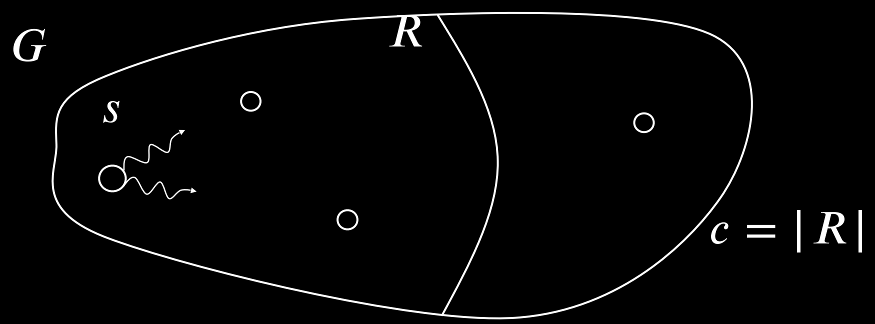
NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$.

c ===> path

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)



NL = coNL (part 2/4) – key idea

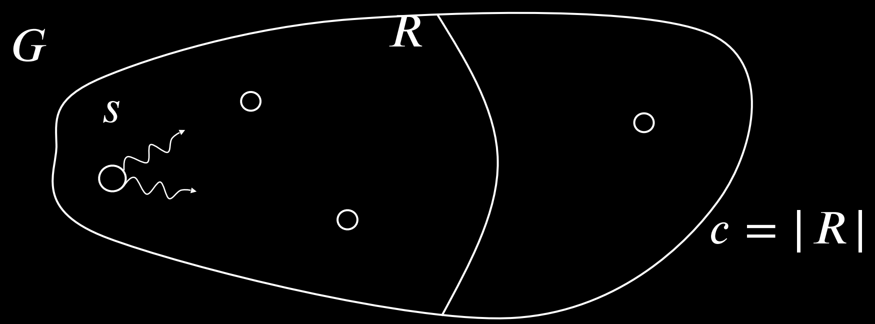
Theorem: If some NL-machine computes c, then some NL-machine computes $path$.

c ===> path

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)

   (p) Nondeterministically pick a path from s to u of length $\leq m$.



NL = coNL (part 2/4) – key idea

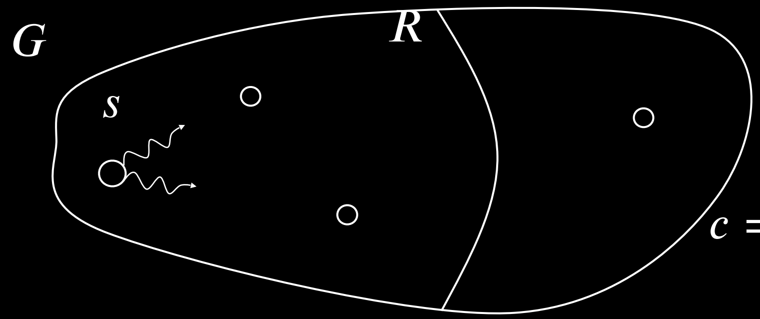
Theorem: If some NL-machine computes c, then some NL-machine computes $path$.

c ===> path

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)

   (p) Nondeterministically pick a path from s to u of length $\leq m$.
   If fail, then *reject*.



$c = |R|$

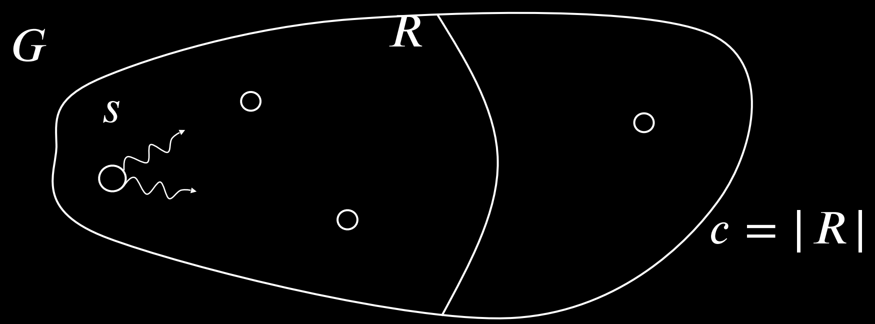
NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$. c ===> path

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)

   (p) Nondeterministically pick a path from s to u of length $\leq m$.
   If fail, then *reject*.
   If $u = t$, then output YES, else set $k \leftarrow k + 1$.

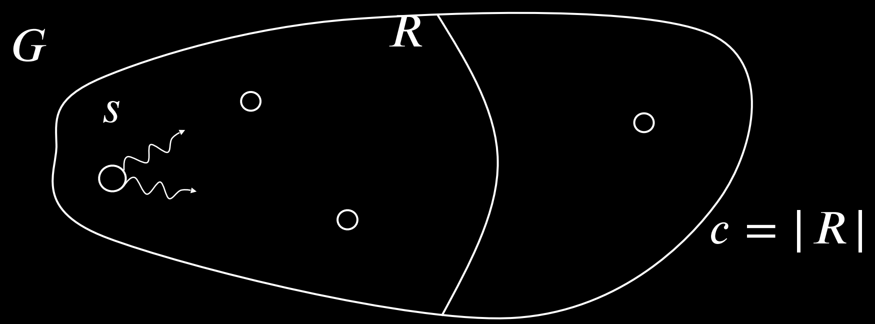


NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$. c ===> path

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   (p) Nondeterministically pick a path from s to u of length $\leq m$.
   If fail, then *reject*.
   If $u = t$, then output YES, else set $k \leftarrow k + 1$.
   (n) Skip u and continue.



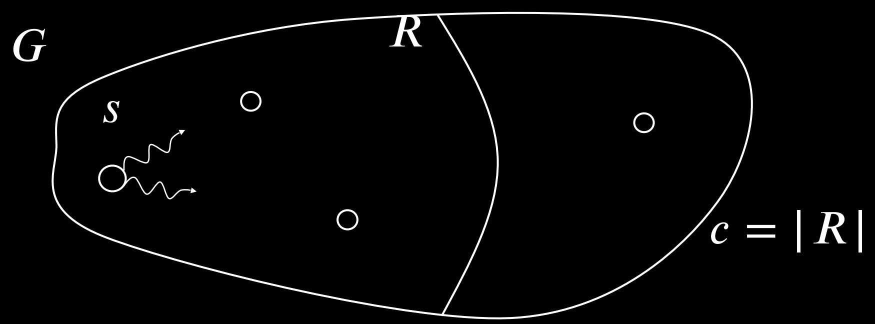
NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$.

c ===> path

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   - (p) Nondeterministically pick a path from s to u of length $\leq m$.
     If fail, then *reject*.
     If $u = t$, then output YES, else set $k \leftarrow k + 1$.
   - (n) Skip u and continue.
5. If $k \neq c$ then *reject*.

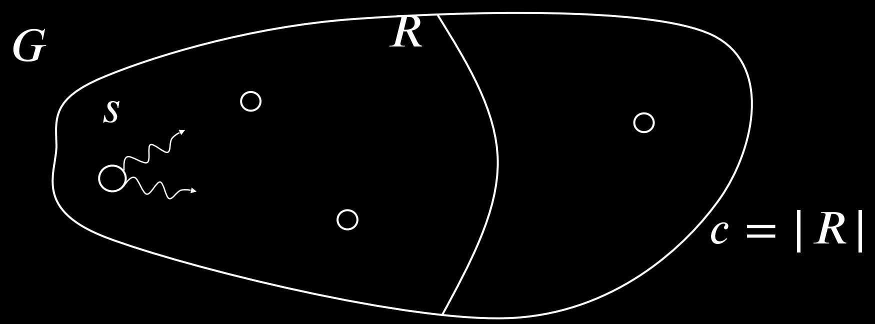


NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$. c ===> path

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   - (p) Nondeterministically pick a path from s to u of length $\leq m$.
     If fail, then *reject*.
     If $u = t$, then output YES, else set $k \leftarrow k + 1$.
   - (n) Skip u and continue.
5. If $k \neq c$ then *reject*.
6. Output NO." [found all c reachable nodes and none were t}

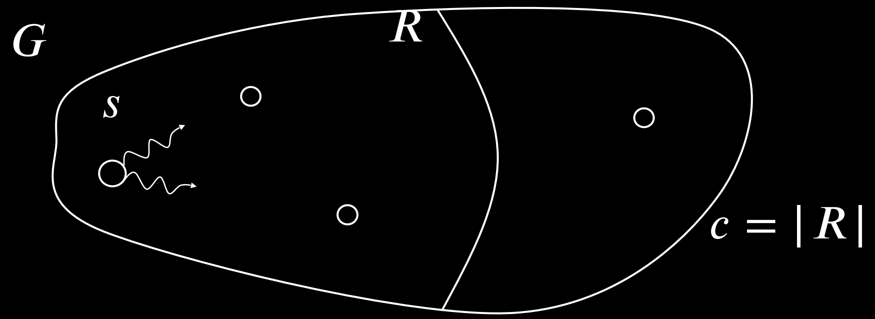


NL = coNL (part 2/4) – key idea

Theorem: If some NL-machine computes c, then some NL-machine computes $path$. c ===> path

Proof: "On input $\langle G, s, t\rangle$

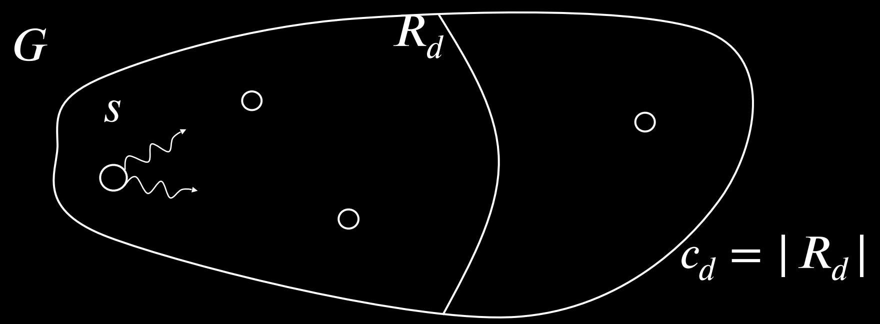
1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   (p) Nondeterministically pick a path from s to u of length $\leq m$.
   If fail, then *reject*.
   If $u = t$, then output YES, else set $k \leftarrow k + 1$.
   (n) Skip u and continue.
5. If $k \neq c$ then *reject*.
6. Output NO." [found all c reachable nodes and none were t}



NL = coNL (part 3/4)

Let $path_d(G, s, t) = \begin{cases} \text{YES, if } G \text{ has a path } s \text{ to } t \text{ of length } \leq d \\ \text{NO, if not} \end{cases}$

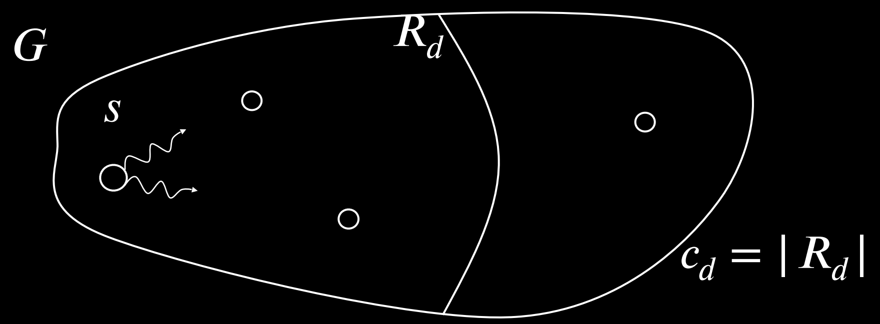
c_d ===> $path_d$



NL = coNL (part 3/4)

Let $path_d(G, s, t) = \{$ YES, if G has a path s to t of length $\leq d$
NO, if not

c_d ===> $path_d$

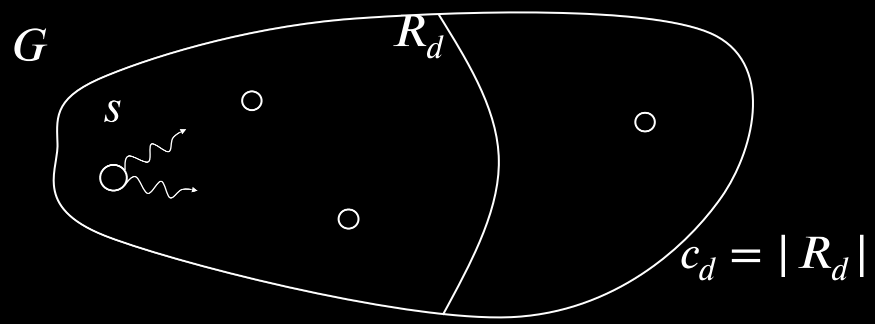


NL = coNL (part 3/4)

Let $path_d(G, s, t) = \{$ YES, if G has a path s to t of length $\leq d$
NO, if not

Let $R_d = R_d(G, s) = \{u \mid path_d(G, s, u) =$ YES$\}$

c_d ===> $path_d$



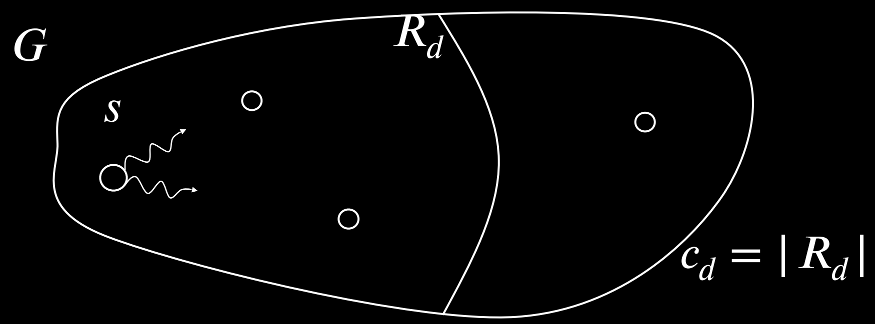
NL = coNL (part 3/4)

Let $path_d(G, s, t) = \{$ YES, if G has a path s to t of length $\leq d$
NO, if not

Let $R_d = R_d(G, s) = \{u \mid path_d(G, s, u) =$ YES$\}$

Let $c_d = c_d(G, s) = |R_d|$

c_d ===> $path_d$



NL = coNL (part 3/4)

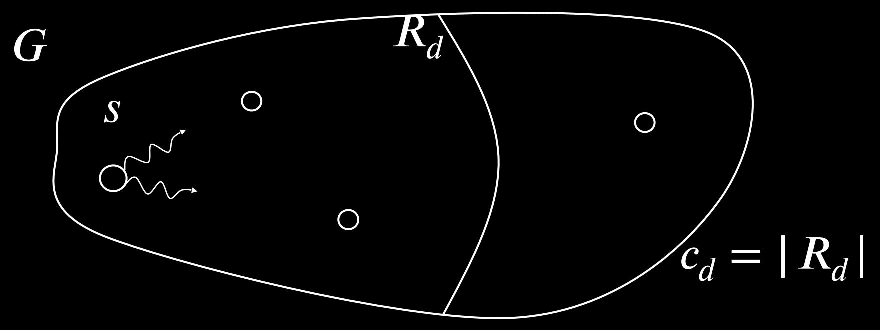
Let $path_d(G, s, t) = \begin{cases} \text{YES}, & \text{if } G \text{ has a path } s \text{ to } t \text{ of length } \leq d \\ \text{NO}, & \text{if not} \end{cases}$

Let $R_d = R_d(G, s) = \{u \mid path_d(G, s, u) = \text{YES}\}$

Let $c_d = c_d(G, s) = |R_d|$

Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_d$.

c_d ===> $path_d$



NL = coNL (part 3/4)

Let $path_d(G, s, t) = \{$ YES, if G has a path s to t of length $\leq d$
NO, if not

Let $R_d = R_d(G, s) = \{u \mid path_d(G, s, u) =$ YES$\}$

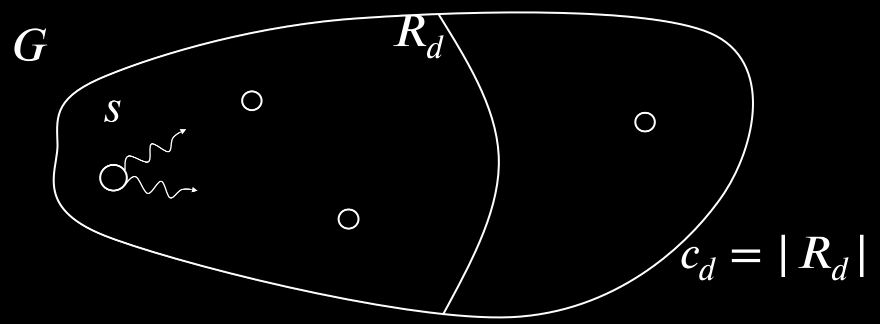
Let $c_d = c_d(G, s) = | R_d |$

Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_d$.

c_d ===> $path_d$

Proof: "On input $\langle G, s, t \rangle$

1. Compute c_d
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   - (p) Nondeterministically pick a path from s to u of length $\leq d$.
     If fail, then *reject*.
     If $u = t$, then output YES, else set $k \leftarrow k + 1$.
   - (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO" [found all c_d reachable nodes and none were t}



NL = coNL (part 3/4)

Let $path_d(G, s, t) = \{$ YES, if G has a path s to t of length $\leq d$
NO, if not

Let $R_d = R_d(G, s) = \{u \mid path_d(G, s, u) =$ YES$\}$

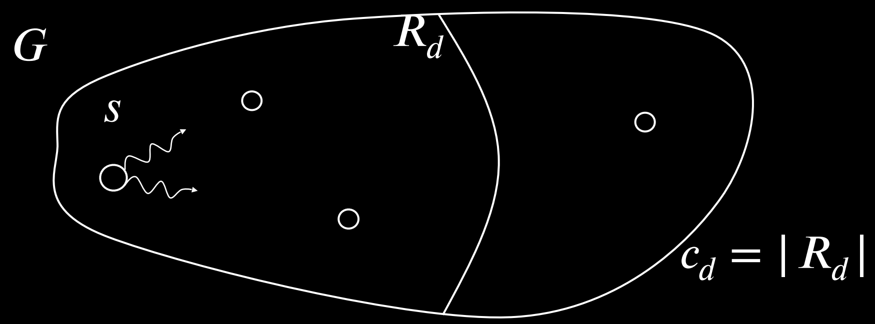
Let $c_d = c_d(G, s) = |R_d|$

Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_d$.

c_d ===> $path_d$

Proof: "On input $\langle G, s, t\rangle$

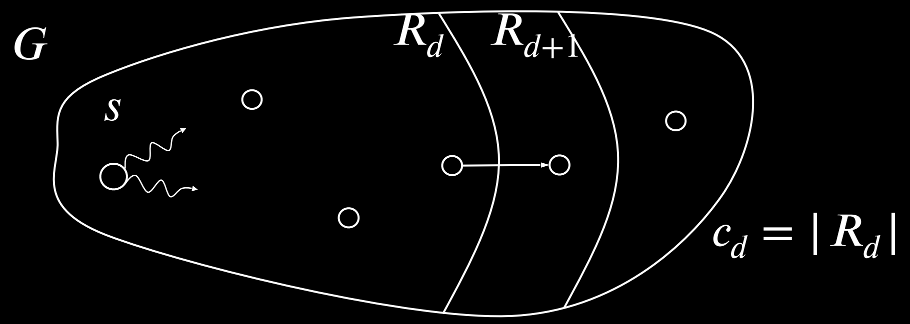
1. Compute c_d
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   (p) Nondeterministically pick a path from s to u of length $\leq d$.
   If fail, then *reject*.
   If $u = t$, then output YES, else set $k \leftarrow k + 1$.
   (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO" [found all c_d reachable nodes and none were t}



NL = coNL (part 4/4)

Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_{d+1}$.

c_d ===> $path_{d+1}$



c_d ===> c_{d+1}

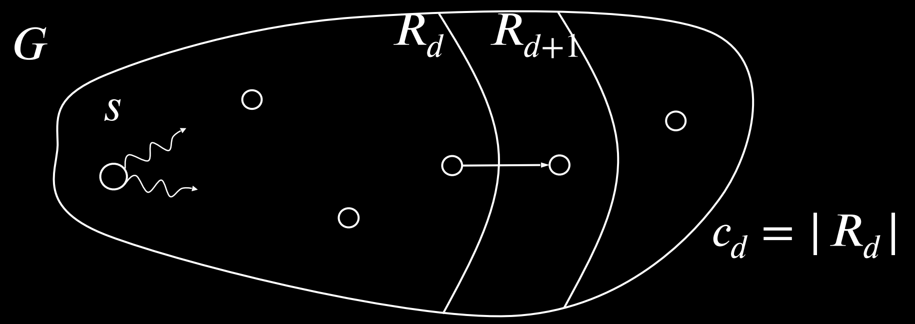
NL = coNL (part 4/4)

Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_{d+1}$.

$c_d \Longrightarrow path_{d+1}$

Proof: "On input $\langle G, s, t \rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   - (p) Nondeterministically pick a path from s to u of length $\leq d$.
     If fail, then *reject*.
     If u has an edge to t, then output YES, else set $k \leftarrow k + 1$.
   - (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO." [found all c_d reachable nodes
   and none had an edge to t}



$c_d \Longrightarrow c_{d+1}$

NL = coNL (part 4/4)

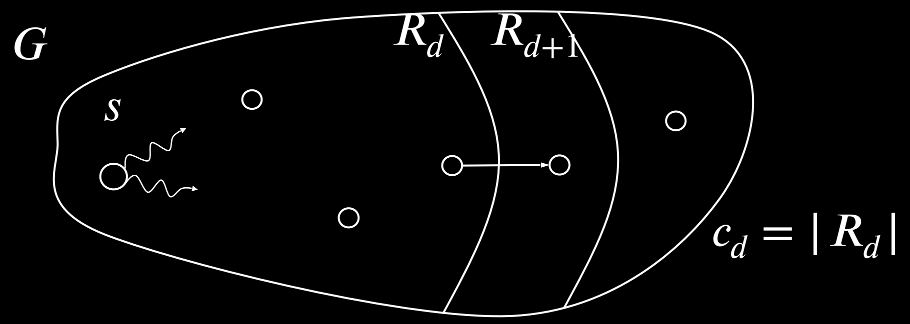
Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_{d+1}$. c_d ===> $path_{d+1}$

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)

   (p) Nondeterministically pick a path from s to u of length $\leq d$.
   If fail, then *reject*.
   If u has an edge to t, then output YES, else set $k \leftarrow k+1$.

   (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO." [found all c_d reachable nodes and none had an edge to t}



Corollary: Some NL-machine computes c_{d+1} from c_d.

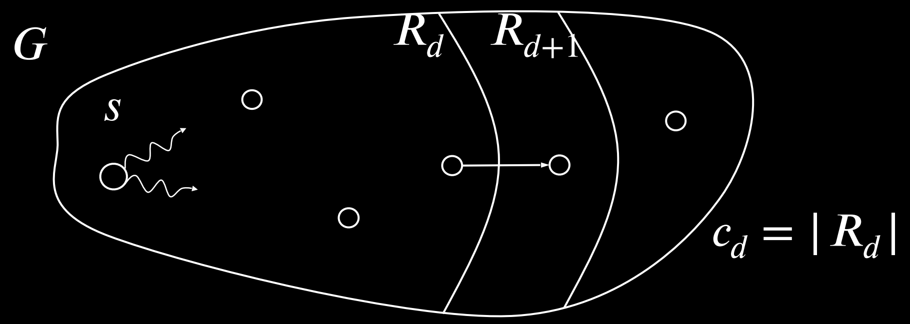
c_d ===> c_{d+1}

NL = coNL (part 4/4)

Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_{d+1}$. c_d ===> $path_{d+1}$

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   (p) Nondeterministically pick a path from s to u of length $\leq d$.
   If fail, then *reject*.
   If u has an edge to t, then output YES, else set $k \leftarrow k+1$.
   (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO." [found all c_d reachable nodes and none had an edge to t}



Hence $\overline{PATH} \in$ NL

Corollary: Some NL-machine computes c_{d+1} from c_d.

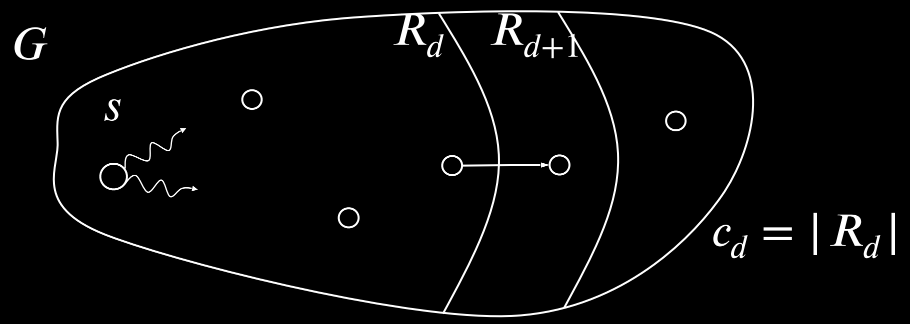
c_d ===> c_{d+1}

NL = coNL (part 4/4)

Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_{d+1}$. c_d ===> $path_{d+1}$

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   (p) Nondeterministically pick a path from s to u of length $\leq d$.
   If fail, then *reject*.
   If u has an edge to t, then output YES, else set $k \leftarrow k+1$.
   (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO." [found all c_d reachable nodes
   and none had an edge to t}



Hence $\overline{PATH} \in$ NL

"On input $\langle G, s, t\rangle$

Corollary: Some NL-machine computes c_{d+1} from c_d.

c_d ===> c_{d+1}

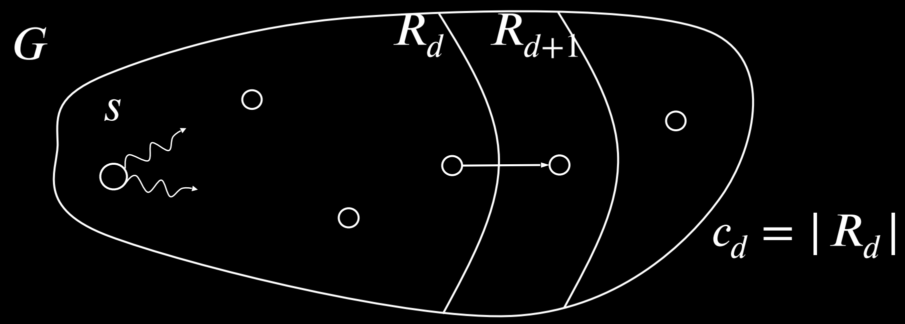
NL = coNL (part 4/4)

Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_{d+1}$.

c_d ===> $path_{d+1}$

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   (p) Nondeterministically pick a path from s to u of length $\leq d$.
   If fail, then *reject*.
   If u has an edge to t, then output YES, else set $k \leftarrow k + 1$.
   (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO." [found all c_d reachable nodes
   and none had an edge to t}



Hence $\overline{PATH} \in$ NL

"On input $\langle G, s, t\rangle$

1. $c_0 = 1$.

Corollary: Some NL-machine computes c_{d+1} from c_d.

c_d ===> c_{d+1}

NL = coNL (part 4/4)

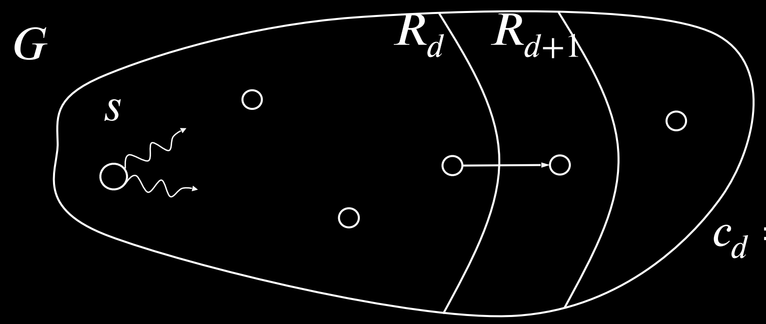
Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_{d+1}$. c_d ===> $path_{d+1}$

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   (p) Nondeterministically pick a path from s to u of length $\leq d$.
   If fail, then *reject*.
   If u has an edge to t, then output YES, else set $k \leftarrow k + 1$.
   (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO." [found all c_d reachable nodes and none had an edge to t}

Corollary: Some NL-machine computes c_{d+1} from c_d.

c_d ===> c_{d+1}



Hence $\overline{PATH} \in$ NL

"On input $\langle G, s, t\rangle$

1. $c_0 = 1$.
2. Compute each c_{d+1} from c_d for $d = 1$ to m.

NL = coNL (part 4/4)

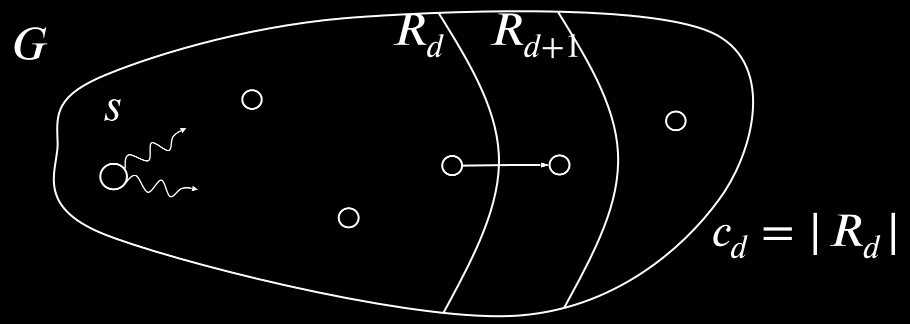
Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_{d+1}$. c

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   - (p) Nondeterministically pick a path from s to u of length $\leq d$.
     If fail, then *reject*.
     If u has an edge to t, then output YES, else set $k \leftarrow k + 1$.
   - (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO." [found all c_d reachable nodes
   and none had an edge to t}

Corollary: Some NL-machine computes c_{d+1} from c_d.

c_d ===> c_{d+1}



Hence $\overline{PATH} \in$ NL

"On input $\langle G, s, t\rangle$

1. $c_0 = 1$.
2. Compute each c_{d+1} from c_d for $d = 1$ to m.
3. *Accept* if $path_m(G, s, t)$ = NO.

NL = coNL (part 4/4)

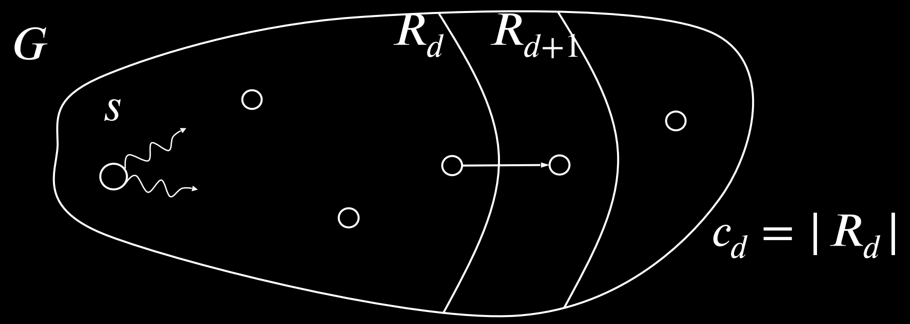
Theorem: If some NL-machine computes c_d, then some NL-machine computes $path_{d+1}$. c_d ===> $path_{d+1}$

Proof: "On input $\langle G, s, t\rangle$

1. Compute c
2. $k \leftarrow 0$
3. For each node u
4. Nondeterministically go to (p) or (n)
   - (p) Nondeterministically pick a path from s to u of length $\leq d$.
     If fail, then *reject*.
     If u has an edge to t, then output YES, else set $k \leftarrow k + 1$.
   - (n) Skip u and continue.
5. If $k \neq c_d$ then *reject*.
6. Output NO." [found all c_d reachable nodes
   and none had an edge to t}

Corollary: Some NL-machine computes c_{d+1} from c_d.

c_d ===> c_{d+1}



Hence $\overline{PATH} \in$ NL

"On input $\langle G, s, t\rangle$

1. $c_0 = 1$.
2. Compute each c_{d+1} from c_d for $d = 1$ to m.
3. *Accept* if $path_m(G, s, t)$ = NO.
4. *Reject* if $path_m(G, s, t)$ = YES."

Quick review of today

1. Log-space reducibility
2. L = NL? question
3. $PATH$ is NL-complete
4. NL = coNL

Quick review of today

1. Log-space reducibility
2. L = NL? question
3. $PATH$ is NL-complete
4. NL = coNL