Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 22, 2019

Lecture 9

NL

Agenda

- about logarithmic space
- paths . . .
- . . . and the absence thereof
- Immerman-Szelepcsényi and others

What can one do with logarithmic space?

In essence an algorithm can maintain a constant number of

- pointers into the input
  - for instance node identities (graph problems)
  - head positions
- counters up to input length

What can one do with logarithmic space?

In essence an algorithm can maintain a constant number of

- pointers into the input
  - for instance node identities (graph problems)
  - head positions
- counters up to input length

Examples:

- **L**: basic arithmetic
- **NL**: paths in graphs

Technical issues

- space usage refers to work tapes only
- read-only input and write-once output is allowed to use more than $\log n$ cells
- write-once: output head must not move to the left
- logspace reductions (because polynomial time-reductions too powerful)

Logspace reductions

Definition (logspace reduction)

Let $L, L' \subseteq \{0,1\}^*$ be languages. We say that L is logspace-reducible to L', written $L \leq_{log} L'$ if there is a function $f : \{0,1\}^* \to \{0,1\}^*$ computed by a deterministic TM using logarithmic space such that $x \in L \Leftrightarrow f(x) \in L'$ for every $x \in \{0,1\}^*$.

- $\leq_{log}$ is transitive
- $C \in \mathbf{L}$ and $B \leq_{log} C$ implies $B \in \mathbf{L}$

- **NL**-hardness and **NL**-completeness defined in terms of logspace reductions

Logspace reductions

Definition (logspace reduction)

Let $L, L' \subseteq \{0,1\}^*$ be languages. We say that L is logspace-reducible to L', written $L \leq_{log} L'$ if there is a function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ computed by a deterministic TM using logarithmic space such that $x \in L \Leftrightarrow f(x) \in L'$ for every $x \in \{0,1\}^*$.

- $\leq_{log}$ is transitive
- $C \in \mathbf{L}$ and $B \leq_{log} C$ implies $B \in \mathbf{L}$
    - Space does not bound time and output size: possibly $|f(w)| \neq \mathcal{O}(\log(|w|))$

- **NL**-hardness and **NL**-completeness defined in terms of logspace reductions

Logspace reductions

Definition (logspace reduction)

Let $L, L' \subseteq \{0,1\}^*$ be languages. We say that L is logspace-reducible to L', written $L \leq_{log} L'$ if there is a function $f : \{0,1\}^* \to \{0,1\}^*$ computed by a deterministic TM using logarithmic space such that $x \in L \Leftrightarrow f(x) \in L'$ for every $x \in \{0,1\}^*$.

- $\leq_{log}$ is transitive
- $C \in \mathbf{L}$ and $B \leq_{log} C$ implies $B \in \mathbf{L}$
  - Space does not bound time and output size: possibly $|f(w)| \neq \mathcal{O}(\log(|w|))$
  - Compute $f(x)$ on demand: store only current symbol and its cell number
- **NL**-hardness and **NL**-completeness defined in terms of logspace reductions

Read-once Certificates

Similar to **NP**, also **NL** has a characterization using certificates

Theorem (read-once certificates)

*$L \subseteq \{0,1\}^*$ is in **NL** iff there exists a det. logspace TM M (verifier) and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \{0,1\}^*$*

$$x \in L \text{ iff } \exists u \in \{0,1\}^{p(|x|)}. M(x,u) = 1$$

Certificate u is written on an additional read-once input tape of M.

Read-once Certificates

Similar to **NP**, also **NL** has a characterization using certificates

Theorem (read-once certificates)

$L \subseteq \{0,1\}^*$ *is in* **NL** *iff there exists a det. logspace TM* M *(verifier) and a polynomial* $p : \mathbb{N} \to \mathbb{N}$ *such that for every* $x \in \{0,1\}^*$

$$x \in L \text{ iff } \exists u \in \{0,1\}^{p(|x|)}.M(x,u) = 1$$

Certificate u is written on an additional read-once input tape of M.

- example: path in a graph is a read-once certificate

⇒ certificate is sequence of choices

⇐ certificate is guessed bit-wise (it cannot be stored)

Agenda

- about logarithmic space ✓
- paths . . .
- . . . and the absence thereof
- Immerman-Szelepcsényi and others

NL is all about paths

Recall the language Path in directed graphs defined as

$$\{\langle G, s, t\rangle \mid \exists \text{a path from } s \text{ to } t \text{ in directed graph } G\}$$

NL is all about paths

Recall the language Path in directed graphs defined as

$$\{\langle G, s, t\rangle \mid \exists \text{a path from } s \text{ to } t \text{ in directed graph } G\}$$

We have seen in Lecture 3 that Path $\in$ **NL** by guessing a path:

- non-deterministic walks on graphs of n nodes
- if there is a path, it has length $\leq n$
- maintain one pointer to current node
- one counter counting up to n

NL is all about paths

Recall the language Path in directed graphs defined as

$$\{\langle G, s, t\rangle \mid \exists \text{a path from } s \text{ to } t \text{ in directed graph } G\}$$

We have seen in Lecture 3 that Path $\in$ **NL** by guessing a path:

- non-deterministic walks on graphs of n nodes
- if there is a path, it has length $\leq n$
- maintain one pointer to current node
- one counter counting up to n

In fact we even have:

Theorem (Path)

Path *is* **NL***-complete.*

Proof

- let $L \in$ **NL** be arbitrary, decided by NDTM M

Proof

- let $L \in$ **NL** be arbitrary, decided by NDTM M
- on input $x \in \{0, 1\}^n$ reduction f outputs configuration graph $G(M, x)$ of size $2^{O(\log n)}$ by counting to n

Proof

- let $L \in$ **NL** be arbitrary, decided by NDTM M
- on input $x \in \{0,1\}^n$ reduction f outputs configuration graph $G(M,x)$ of size $2^{O(\log n)}$ by counting to n
- there exists a path from C_{start} to C_{accept} in $G(M,x)$ iff M accepts x
- path itself can be used as read-once certificate

More path problems

- many natural problems correspond to path (reachability) problems
- the word problem for NFAs: $\{\langle A, w\rangle \mid w \text{ is accepted by NFA } A\}$
- cycle detection/connected components in directed graphs
- $\overline{\text{2SAT}} \in$ **NL**

More path problems

- many natural problems correspond to path (reachability) problems
- the word problem for NFAs: $\{\langle A, w\rangle \mid w \text{ is accepted by NFA } A\}$
- cycle detection/connected components in directed graphs
- $\overline{\text{2SAT}} \in$ **NL**
  - $x \vee y$ equivalent to $\neg x \implies y$ equivalent to $\neg y \implies x$
  - yields an implication graph (computable in logspace)
  - unsatisfiable iff there exists a path $x \to \overline{x} \to x$ in implication graph for variable x

Certificates for absence of paths?

- recall the open problem **NP** $=$ **coNP**?
- equivalent to asking whether unsatisfiability has short certificates
- possibly not

Certificates for absence of paths?

- recall the open problem **NP** $=$ **coNP**?
- equivalent to asking whether unsatisfiability has short certificates
- possibly not

What about absence of paths from s to t in graph G with n nodes named 1,. . . ,n?

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known
    1. list all membership certificates for all $u \in C_i$ sorted in ascending order
    2. check validity and sortedness
    3. check that v is not in the list
    4. check that the list has length $|C_i|$

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known
  1. list all membership certificates for all $u \in C_i$ sorted in ascending order
  2. check validity and sortedness
  3. check that v is not in the list
  4. check that the list has length $|C_i|$
- non-membership in C_i is known given $|C_{i-1}|$ (checking neighbors in (3) as well)

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known
  1. list all membership certificates for all $u \in C_i$ sorted in ascending order
  2. check validity and sortedness
  3. check that v is not in the list
  4. check that the list has length $|C_i|$
- non-membership in C_i is known given $|C_{i-1}|$ (checking neighbors in (3) as well)
- $|C_i| = c$ can be certified given $|C_{i-1}|$ using $C_0 = \{s\}$ as base case

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known
  1. list all membership certificates for all $u \in C_i$ sorted in ascending order
  2. check validity and sortedness
  3. check that v is not in the list
  4. check that the list has length $|C_i|$
- non-membership in C_i is known given $|C_{i-1}|$ (checking neighbors in (3) as well)
- $|C_i| = c$ can be certified given $|C_{i-1}|$ using $C_0 = \{s\}$ as base case

Certificate is certificate for non-membership in C_n!

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known
  1. list all membership certificates for all $u \in C_i$ sorted in ascending order
  2. check validity and sortedness
  3. check that v is not in the list
  4. check that the list has length $|C_i|$
- non-membership in C_i is known given $|C_{i-1}|$ (checking neighbors in (3) as well)
- $|C_i| = c$ can be certified given $|C_{i-1}|$ using $C_0 = \{s\}$ as base case

Certificate is certificate for non-membership in C_n!
Its size is polynomial in number of nodes and read-once!

NL algorithm for $\overline{PATH}$

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

1. Let $c_0 = 1$. ⟦ $A_0 = \{s\}$ has 1 node ⟧
2. For $i = 0$ to $m - 1$: ⟦ compute c_{i+1} from c_i ⟧
3. Let $c_{i+1} = 1$. ⟦ c_{i+1} counts nodes in A_{i+1} ⟧
4. For each node $v \neq s$ in G: ⟦ check if $v \in A_{i+1}$ ⟧
5. Let $d = 0$. ⟦ d re-counts A_i ⟧
6. For each node u in G: ⟦ check if $u \in A_i$ ⟧
7. Nondeterministically either perform or skip these steps:
8. Nondeterministically follow a path of length at most i from s and *reject* if it doesn't end at u.
9. Increment d. ⟦ verified that $u \in A_i$ ⟧
10. If (u, v) is an edge of G, increment c_{i+1} and go to stage 5 with the next v. ⟦ verified that $v \in A_{i+1}$ ⟧
11. If $d \neq c_i$, then *reject*. ⟦ check whether found all A_i ⟧
12. Let $d = 0$. ⟦ c_m now known; d re-counts A_m ⟧
13. For each node u in G: ⟦ check if $u \in A_m$ ⟧
14. Nondeterministically either perform or skip these steps:
15. Nondeterministically follow a path of length at most m from s and *reject* if it doesn't end at u.
16. If $u = t$, then *reject*. ⟦ found path from s to t ⟧
17. Increment d. ⟦ verified that $u \in A_m$ ⟧
18. If $d \neq c_m$, then *reject*. ⟦ check whether found all of A_m ⟧
    Otherwise, *accept*."

NL $=$ coNL

We have just argued the existence of polynomial read-once certificates for absence of paths.

Theorem (Immerman-Szelepcsényi)

NL $=$ coNL.

Further Reading

- paths in undirected graphs is in **L**
  - *Omer Reingold* Undirected ST-Connectivity in Log-Space, STOC 2005
  - available from

`http://www.wisdom.weizmann.ac.il/˜reingold/publications/sl.ps`

- an alternative characterization of **NL** by reachability is at the heart of descriptive complexity
  - **NL** is first-order logic plus transitive closure
  - *Neil Immerman*, Descriptive Complexity, Springer 1999.

What have we learnt?

- space classes closed under complement
  - so are context-sensitive language (see exercises)
- analogous results for time complexity unlikely
- space classes beyond logarithmic closed under non-determinism
- **NL** is all about reachability
- $\overline{\text{2SAT}}$ is in **NL** and thus also 2SAT (in fact, hard for **NL**)
- **NL** has polynomial read-once certificates
- logarithmic space $\sim$ constant number of pointers and counters

Up next: the polynomial hierarchy **PH**