CS:4330 Theory of Computation

Spring 2018

Computability Theory

The Classes L and NL

Haniel Barbosa

THE UNIVERSITY OF IOWA

Readings for this lecture

Chapter 8 of [Sipser 1996], 3rd edition. Section 8.4

Sublinear space bounds

- ▷ We examine smaller, *sublinear* bounds
- ▷ In time complexity sublinear bounds do not even allow reading the entire input, so they are not considered
- ▷ In sublinear space complexity one can read the entire input but not *store* it
- ▷ We consider a different computational model: a TM with a “read-only” input tape and a “read/write” work tape
- ▷ On the read-only tape the head can detect symbols but not change them. The head cannot move beyond the input
- ▷ For at least linear space bounds, the two-tape TM model is equivalent to the standard one-tape model
- ▷ For sublinear space bounds, we use only the two-tape model

L and NL classes

Definition

L is the class of languages that are decidable in logarithmic space on a DTM with a read-only input tape, i.e.

$$\mathbf{L} = \mathrm{SPACE}(\log n)$$

NL is the class of languages that are decidable in logarithmic space on a NTM with a read-only input tape, i.e.

$$\mathbf{NL} = \mathrm{NSPACE}(\log n)$$

- ▷ We focus on $\log n$ space instead of other sublinear spaces for similar reasons to our selection of polynomial bounds
- ▷ Logarithmic space is sufficient for many interesting computational problems and has attractive mathematical properties

Example

$A = \{0^k 1^k \mid k > 0\}$ is a member of $\mathbf{L}$.

- ▷ Zig-zag decision algorithm (linear space) can be modified to use two tapes: a read-only input tape and a read-write work tape

Example

$A = \{0^k 1^k \mid k > 0\}$ is a member of $\mathbf{L}$.

- ▷ Zig-zag decision algorithm (linear space) can be modified to use two tapes: a read-only input tape and a read-write work tape

- ▷ The algorithm counts number of 0s and, separately, number of 1s in the input and write their binary expressions on the work tape

Example

$A = \{0^k 1^k \mid k > 0\}$ is a member of $\mathbf{L}$.

- ▷ Zig-zag decision algorithm (linear space) can be modified to use two tapes: a read-only input tape and a read-write work tape

- ▷ The algorithm counts number of 0s and, separately, number of 1s in the input and write their binary expressions on the work tape

- ▷ Since only space for two binary numbers is required and since number of binary digits in n is $\log n$ we conclude $A \in \mathbf{L}$

Another example

$PATH = \{\langle G, s, t\rangle \mid G$ is a directed graph that has a directed path from s to $t\}$ is a member of $\mathbf{NL}$.

- ▷ Let T be the NTM recognizing *PATH*.
- ▷ The NTM T needs only to remember one node on its work tape.
- ▷ Let that node be x.

Another example

$PATH = \{\langle G, s, t\rangle \mid G$ is a directed graph that has a directed path from s to $t\}$ is a member of $\mathbf{NL}$.

- ▷ Let T be the NTM recognizing *PATH*.
- ▷ The NTM T needs only to remember one node on its work tape.
- ▷ Let that node be x.

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

1. Initially, $x = s$
2. If x has no successors, T *rejects*
3. T then nondeterministically chooses a successor y of x
4. If y is t, then T *accepts*
5. Replace x by y, go to 2."

Log space transducers

A log space transducer M 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 $\mathcal{O}(\log n)$ symbols
- ▷ M computes a function $f : \Sigma^* \to \Sigma^*$, where $f(w)$ is the string remaining on the output tape after M halts when it is started with w on its input tape.
- ▷ f is called a *log space computable function*

Definition (Log space reducibility)

A language A is *log space reducible* to language B, written $A \leq_L B$, if there exists a log space computable function f, such that for any $w \in \Sigma^*$, $w \in A$ iff $f(w) \in B$.

NL-completeness

Definition

A language B is *NL-complete* if:

1. $B \in \mathbf{NL}$
2. For every $A \in \mathbf{NL}$, $A \leq_L B$

Theorem

If $A \leq_L B$ *and* $B \in \mathbf{L}$ *then* $A \in \mathbf{L}$

Corollary

If any NL-complete language is in $\mathbf{L}$ *then* $\mathbf{L} = \mathbf{NL}$

Example

PATH is *NL-complete*.

- ▷ Necessary to show that $PATH$ in *NL-hard*

- ▷ Analogously to previous proofs, one needs to build a graph that represents the computation of the nondeterministic log space Turing machine for A.

Summary

Lemma

$\mathbf{NL} \subseteq \mathbf{P}$

Proof

$PATH \in \mathbf{P}$ and any language in $\mathbf{NL}$ is polynomially time reducible to *PATH*.

In summary

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NSPACE} \subseteq \mathrm{EXPTIME}$$