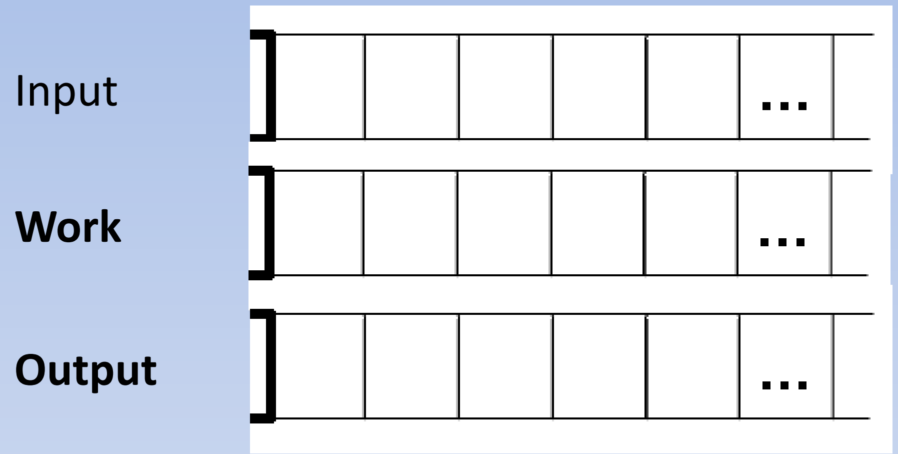
# The classes **L** and **NL**, Savitch's, Immerman-Szelepscényi, Reingold's Theorems

Christos Moyzes
MPLA 2012-2013

# Quick Reminders / Definitions

Consider a Turing Machine with 3 tapes




A problem is in SPACE(s(n)) **iff** a TM uses s(n) space for <u>work</u> and <u>output</u> tapes
A problem is in NSPACE(s(n)) **iff** a NTM uses s(n) space for <u>work</u> and <u>output</u> tapes

L = SPACE ( O(logn) ) NL = NSPACE( O(logn) )

n = the length of input x

# <u>Theorem</u> If a machine halts, and uses space $s(n) \geq \log n$, it runs in time $2^{O(s(n))}$

- configuration : the specific state, position of header(s), contents of tape(s)
- So number of *possible* configurations is states (finite) * length of input x * possible (binary) strings of length s(n) **hence** $O(1)*n*2^{s(n)} = 2^{O(s(n))+\log n} = 2^{O(s(n))}$
- It takes 1 step to visit every state and that is only once (otherwise machine would not halt) so #configurations = time = $2^{O(s(n))}$

# Theorem NL ⊆ P

- It still holds #configurations = $2^{O(s(n))} = 2^{O(\log n)} = n^{O(1)}$
- Consider directed graph with
  vertices = configurations
  edges = allowable transitions

  Configuration Graph
- Question is : Starting from original state is there an acceptance state ?
- We run algorithm for REACHABILITY for each such acceptance “vertex”
- There are polynomially many destination vertices and REACHABILITY is solved polynomially so we are still in P.

# What about reductions?

- For P,NP we reduce in polynomial time

Should we do the same for L,NL ? **NO** because **NL** $\subseteq$ **P**

We will be reducing problems in log space

- Suppose $M_f$ computes f

Problem: If $M_f$ computes f and output $|f(x)| > \log n$ how is $M_f$ working in log space?

Solution: We will ask $M_f$ for only 1 bit at a time.

Definition : A is log space reducible to B, $A \leq_{\log} B$ **iff** $\exists$ a function f implicitly computable in log space such that $x \in A \Leftrightarrow f(x) \in B$

**Theorem** If $A \leq_{\log} B$ and $B \in \mathbf{L} \Rightarrow A \in \mathbf{L}$

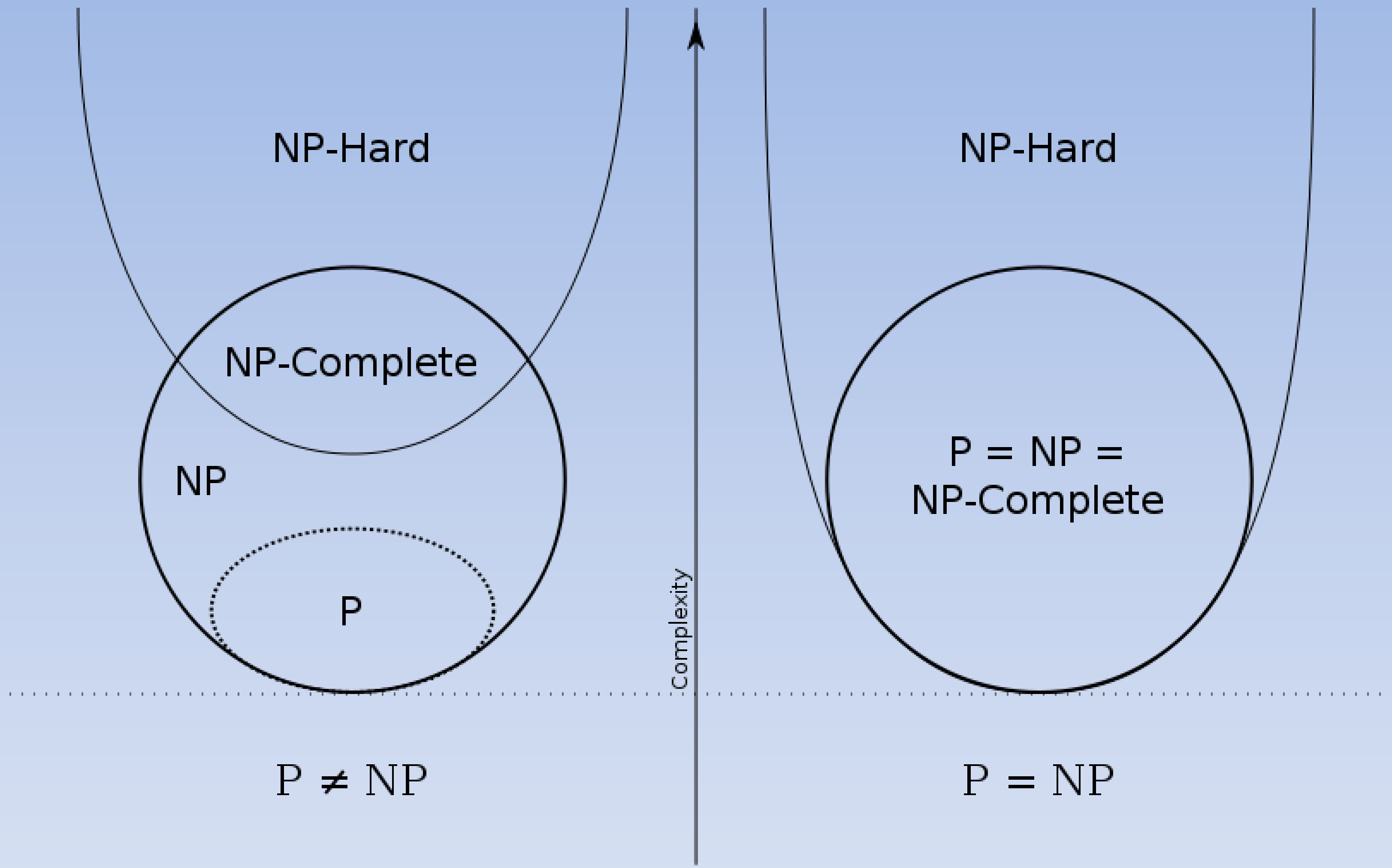
- $M_B$ wants to read bit i of f(x) and "asks" $M_f$
- $M_f$ computes the bit in log space and writes it on output tape.
- repeat

**Definition**

A is NL-Hard if $\forall$ B $\in$ NL, $B \leq_{\log} A$

A is NL-Complete **iff** A is NL-Hard & A $\in$ NL

# The P-NP analog




$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP$

# REACHABILITY(s,t) is NL-Complete

Reachability $\in$ NL

- For every vertex, starting at s, non-deterministically choose neighbor to go to.
- If you reach t in at most n steps then YES
- If you don't reach t in n steps then NO
- We only need to remember the index of the vertex and the number of steps so far. They are numbers at most n, so 2*logn space to represent, hence O(logn)

Reachability is NL-hard

- Let $A \in NL$ ,$M_A$ the machine that decides it, x input of $M_A$. We compute (implicitly) the configuration graph of $M_A(x)$.
- We add a vertex t and add edges from all accepting vertexes to t.
- $M_A(x)$ accepts $\Leftrightarrow$ REACHABILITY(s,t) returns YES
- The reduction f is in log space:
  f need only answer for two vertices at a time so O(logn) space for them.
  Also a sub-routine than can check if transition is allowable is relatively easy.

# Theorem (Savitch) $NSPACE(f(n)) \subseteq SPACE(f^2(n))$

- Let $A \in NSPACE(f(n))$. The configuration graph $G(V,E)$ has $2^{O(f(n))}$ vertices.
- Deterministic recursive algorithm:
  Reach(s, t, 1): $(s, t) \in E$
  Reach(s, t, k): $\forall w \in V \setminus \{s, t\}$ compute
  Reach(s, w,[k/2]) and Reach(w, t,[k/2])
  If they both accept, accept. Else, reject.
- Space: $S(1) = O(f(n))$ to remember what edge we're checking
  $S(k) = O(f(n))$ to remember w + $S(k/2)$
  $\Rightarrow S(k) = O(f(n)) * \log k = O(f(n)) * O(f(n)) = O(f^2(n))$

- This is still the one with the best known space bound.
- Time: $T(1) = O(\ |V|+|E|\ )$
  $T(k) = O(1) + n*2*T(k/2)$
  This solves to $T(k) = n^{O(\log k)}$ (superpolynomial)
- No known algorithm achieves polynomial log-space and polynomial time simultaneously

<u>Corollaries</u>:

- REACHABILITY $\in$ SPACE($\log^2 n$)
- NPSPACE = PSPACE
- Non-determinism is less powerful with respect to space than to time.

# Theorem (Immerman–Szelepcsényi) NSPACE = coNSPACE

Based on NL=coNL

- REACHABILITY' is coNL-complete
- We need to show REACHABILITY' $\in$ NL then we have coNL $\subseteq$ NL
- Similarly REACHABILITY' $\in$ NL $\Rightarrow$
  $\Rightarrow$ REACHABILITY $\in$ co NL
  $\Rightarrow$ NL $\subseteq$ coNL

Idea for REACHABILITY' $\in$ NL

- Algorithm 1
  <u>input</u>: $G = (V, E)$, $s$, $t$, $r$
  <u>output</u>: YES if it discovers that $t$ is not reachable from $s$, and NO otherwise
  <u>assumption</u>: there are exactly $r$ distinct vertices reachable from $s$
- Algorithm 2 (find $r$)
  <u>input</u>: $G = (V,E)$, $s$, $k$, $r_{k-1}$
  <u>output</u>: the number of vertices reachable from $s$ in at most $k$ steps (including $s$ in this count)
  <u>assumption</u>: $r_{k-1}$ is the exact number of vertices reachable from $s$ in at most $k - 1$ steps

# Certificate Definition of NL

We can copy the certificate definition for NP.

Problem : If certificate $p(x)$ is polynomially long, log space work tape can't hold it.

Solution : We allow extra "read once" input tape

Definition 2  $A \in NL$ **iff** there exists log space TM M such that  $x \in A \Leftrightarrow \exists u \ \ |u| < p(|x|) \ \& \ M(x, u) = YES$

- u is on "read once" input tape
- p polynomial

Let N be the NTM of Definition 1

- Definition 1 $\Rightarrow$ Definition 2
  N makes polynomially many guesses
  These guesses form certificate u
  M simulates N and reads guess from tape u
  One u exists that returns YES.

- Definition 2 $\Rightarrow$ Definition 1
  N chooses non-deterministic next bit of u and simulates M.
  Only one bit of u at time (can’t store whole u)

# The class SL

Definition  SL is the class of problems log-space reducible to undirected REACHABILITY
(or can be solved by symmetric NTM)

Theorem (Reingold - 2004)  SL = L

Obvious Consequence  SL-complete problems can be used for the design of log space, polylog space, log reductions.

## SL- complete problems

- USTCON
- Simulation of symmetric Turing machines: does an STM accept a given input in a certain space, given in unary?
- Vertex-disjoint paths: are there *k* paths between two vertices, sharing vertices only at the endpoints? (a generalization of USTCON, equivalent to asking whether a graph is *k*-edge-connected)
- Is a given graph a bipartite graph, or equivalently, does it have a graph coloring using 2 colors?
- Do two undirected graphs have the same number of connected components?
- Does a graph have an even number of connected components?
- Given a graph, is there a cycle containing a given edge?
- Do the spanning forests of two graphs have the same number of edges?
- Given a graph where all its edges have distinct weights, is a given edge in the minimum weight spanning forest?
- Exclusive or 2-satisfiability: given a formula requiring that $x_i$ xor $x_j$ hold for a number of pairs of variables $(x_i,x_j)$, is there an assignment to the variables that makes it true?

# Sources


- Lecture Notes on Computational Complexity, Luca Trevisan
- Computational Complexity: A Modern Approach , Sanjeev Arora and Boaz Barak
- Computational Complexity, Christos Papadimitriou
- http://en.wikipedia.org/wiki/SL_(complexity), Wikipedia