

Summary: Lecture 4

Summary for the chapter 7.3 from page 150 on. [3]

Nondeterministic Turing Machine

A nondeterministic Turing machine (*NTM*) has states, which have more than one possible next state for an action. The states are not completely determined by its action and the current symbol it sees, (unlike a deterministic Turing Machine).

NTMs are for example used in thought experiments. One of the most important problems is the P versus NP problem: How difficult it is to simulate nondeterministic computation with a deterministic computer? [1, 3]

Asymmetry of non-determinism

Asymmetry of nondeterministic acceptance:

- Example: find out if a formula φ is satisfiable ($\varphi \in SAT$):
 - choose truth values for the variables nondeterministically
 - check if they make φ become true
- this approach seems to be unpractical so check whether φ is not satisfiable ($\varphi \in \overline{SAT}$)
- Question whether $NP = coNP$ is a statement about *all* options

TODO with book

Asymmetry of nondeterministic space:

- Example: REACHABILITY $\in NL$
 - starting at start node 1
 - algorithm walks through nondeterministically chosen edges $\leq n$ times
 - only current position is remembered ($\log n$ space)
 - accepts if current node is node n
- this approach seems to be unpractical so check if node n is *not* reachable from node 1

TODO with book

$\log n$ space

A graph algorithm using $O(\log n)$ space stores a fixed number of pointers, independent of n , and manipulates them in some way. [2]

Nondeterministically computing functions

- A nondeterministic Turing Machine M computes a function f if the following hold for every input x :
 - one of the computations of M stops in the halting state h with the correct result $f(x)$ on the output tape
 - all computations that do not correctly output $f(x)$ stop instead in a *no*-state (this path failed then)

TODO with book

Questions:

Does this lead to the Halting problem?

Immerman-Szelepscènyi

Theorem:

Given a graph G and a start node x , the number of nodes that are reachable from x in G can nondeterministically be computed in space $\log n$ (where n is the number of nodes of G).

Proof concept:

- nodes $1, \dots, n$ with start node 1
- $S(i)$ is the set of nodes which are reachable from the startnode with a pathlength of i
 - $S(0)$ will contain node 1
 - $s(1)$ will contain all neighbours of 1
- Algorithm consists out of 3 nested for-loops:
 - outer for loop:
 - * computes number of nodes reachable from initial node (for loop with k steps)
 - * in each step the previous set is overwritten with the next one because the space is limited
 - second for loop:
 - * it is computed how far the previous steps got and summed up how far it can get
 - third loop:
 - * checking if node belongs to $S(i)$
 - * return no when all guesses were correct?
- nodes can't be marked (this would use linear space) but with determinism it gets into $\log n$

Algorithm (2) slides seems to be important

TODO and TODO with book

Questions:

REACHABILITY \in NL

NL = nondeterministic logarithmic space

little bit of stuff between $\log n$ and constant but no interesting stuff

TODO

Questions:

NSPACE is closed under complement

$$\text{NSPACE}(f(n)) = \text{coNSPACE}(f(n))$$

for all proper complexity functions $f(n) \geq \log n$.

TODO

Questions:

References

- [1] Jeff Erickson. *Nondeterministic Turing Machine*. <http://jeffe.cs.illinois.edu/teaching/algorithms/models/09-nondeterminism.pdf>. 2016.
- [2] *Logarithmic Space and NL-Completeness*. http://www.cs.toronto.edu/~ashe/logspace_handout.pdf. 2020.
- [3] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.