Pina Kolling
piko0011

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 in is the P versus NP problem: How difficult it is to simulate nondeterministic computation with a deterministic computer? [1, 3]

Asymmetry of non-deterministism

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 nondeterminstically 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 nondeterminstic Turing Mashine M computes a function f if the the following hold for every input x:
  - one of the computations of M stops in the halting state $h$with the corret 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 Haltingproblem?

Pina Kolling
piko0011

Problem: GRAPH REACHABILITY

Given a graph G and two nodes $n_1, n_2 \in V$, is there path from n_1 to n_2?
A graph $G = (V, E)$ is a finite set V of nodes and a set E of edges as node pairs.

REACHIBILITY can be nondeterministically solved in space $\log n$.

Immerman-Szelepscènyi

Theorem (Counting problem):
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).

- can be non-deterministically solved
- solving as an extension to REACHABILITY
- counting of nodes that can *not* be reached similar: substract result from n
  $\rightarrow$ counting problem and its complement are identical

Algorithm:

- nodes $1, ..., n$ 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 consitsts out of 4 nested for-loops:
  - outer for loop:
    * computes number of nodes reachable from initial node (for loop with k steps) as $|S(1)|, |S(2)|, ..., |S(n-1)|$
    * $|S(n-1)|$ is the desired answer (n is the number of nodes)
    * $|S(0)| = 1$ (contains only start node)
    * $|S(k)|$ is computed after producting $|S(k-1)|$
    * 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
    * a counter l is initialized to 0
    * l gets incremented for each node u which is in $S(k)$ (in the end: $l = |S(k)|$)
  - third loop:
    * deciding if node u belongs to $S(k)$
    * iterating over all nodes $v \in V$ one by one to reuse space
    * if node v is in $S(k-1)$, a counter m is incremented
      m counts the members of $S(k-1)$ that were found so far
    * if $u = v$ or there is an edge from u to v: $u \in S(k)$
      $\rightarrow$ variable *reply* gets set to true

    - if end is reached:
    - $u \notin S(k)$ if end is reached and reply is false:
      if $m < |S(k-1)|$ not all members of $S(k-1)$ have been ecountered: return *no*
    - else return *reply*

  - fourth loop:
    - checking whether $v \in S(k-1)$ with non-determinism (similar to REACHBILITY)

- nodes can't be marked (this would use linear space)
- runs in space $\log n$ with a Truring Machine M
- M has seperate strings holding each of the variables: $k, |S(k-1)|, l, u, m, v, p, w_p, w_{p-1}, input, output$
  all of those need only to be compared to each other and incremented by 1
  all bounded by n

$\overline{\text{REACHABILITY}} \in$ NL

- NL = nondeterministic logarithmic space
- $\overline{\text{REACHABILITY}} \in$ NL

Modify the non-deterministic Turing Mashine from above so that it returns *yes* if the innermost subroutine ever reaches the target node n, otherwise, return *no*.

- run previous algorithm
- if target node is found, yes is returned
- else algorithm continues

Questions:
Did I understand this right?

NSPACE is closed under complement

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

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

Proof idea:

- Language $L \in$NSPACE($f(n)$) is decided by an non-deterministic Turing Mashine M
- M is space bounded in $f(n)$
- to show: there is a $f(n)$ space bounded non-deterministic Turing Mashine $\overline{M}$ which decides $\overline{L}$:
  - On input x $\overline{M}$ runs the recursive algorithm of the *Savitch-Theorem-proof* (choses internal node on the middle) on the configuration graph of M
  - the algorithm decides if two nodea are connected on the basis of x and the transition function of M
  - if $\overline{M}$ comes to an accepting configuration U, it halts and rejects
  - otherwise (if it is computed and no accepting configuration has been found) $\overline{M}$ accepts

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.