

Computational Complexity Theory

Course Notes

September 20, 2006

Lecture 5 - september 13, 2006

Problem from last time (open \rightarrow not open)

$\exists K \in \text{DTIME}(n^{1000})$ so that $\forall m$, m decides L , $\forall x$ (except finitely many): M uses time $\geq n^2$ on x .

This problem is NOT, as claimed last time, OPEN:

Proof:

$$\text{DTIME}(n^{1000}) = t'(n), n^2 = t(n)$$

Inputs: $x_0 = \epsilon, x_1 = 0, x_2 = 1, x_3 = 01, \dots$

Machines: m_0, m_1, m_2, \dots

Procedure for L

- Input (x_i)
- (* Find $D(i)$ and make sure that:
$$x_i \in L \quad \Leftrightarrow \quad x_i \notin L (M_{D(i)})$$
 *)
- Let $D(i)$ be the smallest j so that:
 1. M_j uses time $< t(|x_i|)$ on input x_i
 2. $D(i) \notin \{D(1), D(2), D(3), \dots, D(i-1)\}$
 - (Search for the smallest that bother us.)

It takes exponential time because of the check in 2. (We check because we don't want to diagonalize against the same machine again.)

What we want: All problem machines must be handled eventually.

Let $g(h)$ be a function so that: $g(n) \uparrow g(n) \rightarrow \infty$ as $n \rightarrow \infty$. BUT very slowly. Something like:

$$g(n) = \log^*(n) := \min \{j \mid \log^j(n) < 1\}$$

This means changing 2. to:

$$D(i) \notin \{D(1), D(2), D(3), \dots, D(g(n))\}$$

- If no such j exist accept j , otherwise:
 1. Accept if $M_{D(i)}$ rejects x_i
 2. Reject if $M_{D(i)}$ accepts x_i

Exercise:

What is $\{L \mid \exists L' \in \text{DSpace}(\sqrt{n}) : L \leq_{\log} L'\}$

Let

- $L \in \text{DSpace}(n)$
- $L' = \{ \langle x, 0|x|^2 \rangle \mid x \in L \}$
- $L' \in \text{DSpace}(\sqrt{n})$
- $L \rightarrow L'$
- $x \xrightarrow{r} \langle x, 0|x|^2 \rangle$

Solution: PSPACE
 (Because L could belong to $\text{DSpace}(n^{O(1)})$)

Exercise:

$\text{NP} \neq \text{DTIME}[2^{O(n)}], 2^{O(n)} = \text{E}$

(DTIME is semi-robust)

Proof: NP is downward closed under \leq_{\log}
 DTIME $[2^{O(n)}]$ is not downward closed under \leq_{\log}
 Closed DTIME $(2^{O(n)}) = \text{EXP}$

Immerman – Szelepcényi theorem ('87)

$\text{NSPACE}[O(S(n))] = \text{coNSPACE}[O(S(n))]$
 For $s(n) \geq \log(n)$ and $S(i)$ spaceconstructible.

In particular $\text{NL} = \text{coNL}, \text{NSPACE}[O(n)]^*$

Immer-Szel-proof:

By upwards translation it is sufficient to show $\text{NL} = \text{coNL}$.

And by NL completeness of GAP^\dagger it is sufficient to show that $\text{GAP} \in \text{coNL} \Leftrightarrow \overline{\text{GAP}} \in \text{NL}$.

$\overline{\text{GAP}} =$

\Leftrightarrow Given graph G we want a nondeterministic program which can accept
 there is no path from 1 to N in G.

```

nonGAP() {
  c ← NumberOfNodesReachableFromOne()
  a := 0
  for ( v := 1 to n - 1 )
  {

```

* NSPACE = context sensitive languages.

† Graph accessibility Problem.

```

    Guess a path from 1 to I of length  $\leq n$ .
    if guessed then d++
  }
  if (d == c)
    then accept
    else reject
}

```

```

NumberOfNodesReachableFromOne()‡ {
  d := 1
  i := 1
  //Invariant: d is the number of nodes reachable from 1 in at most i steps.
  repeat {
    d := inductiveCounting (d, i)
    i ++
  } until (i == n)
}

```

```

InductiveCounting (d, i) {
  c := 0
  for ( v := 1 to n )
    if ( IsSubPath (v, i, d) ) {
      c++
    }
  return c
}

```

```

IsSubPath (v, i, d) {
  c := 0
  for (u  $\in$  {1, 2, ..., n} - {v} ) {
    guess a path from 1 to u of length at most i
    if (successful) then c++
  }
}

```

```

guess a path from 1 to v
if (successful)
  then k := 1
  else k = 0
if (d  $\neq$  c + k) reject //We are somewhere where we don't know where we are.
if ( k == 0) { // return k != 0
  then return false
  else return true
}
}
}

```

$L \subseteq NL (= coNL) \subseteq P \subseteq NP, coNP \subseteq PSPACE \subseteq EXP \subseteq NEXP$

[‡] Uses *Inductive counting*

