NTIME Hierarchy Theorem

Theorem (Nondeterministic Time Hierarchy Theorem):
For every proper complexity function \( f(n) \geq n \), and \( g(n) = \omega(f(n+1)) \),

\[
\text{NTIME}(f(n)) \subsetneq \text{NTIME}(g(n)).
\]
NTIME Hierarchy Theorem

• Did we diagonalize against $M_i$?
  – if $L(M_i) = L(D)$ then:
    
    $M_i$ : \[1^n, y, 2, 2, 2, \ldots, 2\]
    $D$ : \[1^n, 1, 1, 1, \ldots, n\]
  
  – equality along all arrows.
  – contradiction.

NTIME Hierarchy Theorem

• General scheme:
  – interval $[1\ldots t(1)]$ kills $M_1$
  – interval $[t(1)\ldots t(t(1))]$ kills $M_2$
  – interval $[t^{i-1}(1)\ldots t(1)]$ kills $M_i$

  • Running time of $D$ on $1^n$: $f(n+1) +$ time to compute interval containing $n$

  • conclude $D$ in $\text{NTIME}(g(n))$ \((g(n) = \omega(f(n+1)))\)

Ladner’s Theorem

• Assuming $P \neq \text{NP}$, what does the world (inside $\text{NP}$) look like?

  $\text{NP}$: \[
P, \quad \text{NPC}
  \]

Ladner’s Theorem

• Can enumerate (TMs deciding) all languages in $P$.
  – enumerate TMs so that each machine appears infinitely often
  – add clock to $M_i$ so that it runs in at most $n^i$ steps

Ladner’s Theorem

• Can enumerate (TMs deciding) all $\text{NP}$-complete languages.
  – enumerate TMs $f$, computing all polynomial-time functions
  – machine $N_i$ decides language SAT reduces to via $f$, if $f$ is reduction, else SAT (details omitted…)

Ladner’s Theorem

Theorem (Ladner): If $P \neq \text{NP}$, then there exists $L \in \text{NP}$ that is neither in $P$ nor $\text{NP}$-complete.

• Proof: “lazy diagonalization”
  – deal with similar problem as in NTIME Hierarchy proof
Ladner’s Theorem

- Our goal: \( L \in \text{NP} \) that is neither in \( \text{P} \) nor \( \text{NP} \)-complete

Ladner’s Theorem

- Top half, assuming \( \text{P} \neq \text{NP} \):
  - focus on \( M_i \)
  - for any \( x \), can always find some \( z \geq x \) on which \( M_i \) and \( \text{SAT} \) differ (why?)

Ladner’s Theorem

- Bottom half, assuming \( \text{P} \neq \text{NP} \):
  - focus on \( N_i \)
  - for any \( x \), can always find some \( z \geq x \) on which \( N_i \) and \( \text{TRIV} \) differ (why?)

Ladner’s Theorem

- General scheme: \( f(n) \) slowly increasing function
  - \( f(0) = 0 \), \( f(n) = f(n-1) + 1 \)
  - \( f(0) \) even: answer \( \text{SAT}(x) \)
  - \( f(0) \) odd: answer \( \text{TRIV}(x) \)
  - notice choice only depends on length of input... that’s OK

Ladner’s Theorem

- 1st attempt to define \( f(n) \)
- “eager \( f(n) \)”: increase at 1st opportunity
- Inductive definition: \( f(0) = 0 \)
  - if \( f(n-1) = 2i \), trying to kill \( M_i \)
    - if \( \exists z < 1^n \) s.t. \( M_i(z) \neq \text{SAT}(z) \), then \( f(n) = f(n-1) + 1 \)
  - if \( f(n-1) = 2i+1 \), trying to kill \( N_i \)
    - if \( \exists z < 1^n \) s.t. \( N_i(z) \neq \text{TRIV}(z) \), then \( f(n) = f(n-1) + 1 \)
Ladner’s Theorem

- Problem: eager $f(n)$ too difficult to compute
- on input of length $n$,
  - look at all strings $z$ of length $< n$
  - compute $\text{SAT}(z)$ or $N_i(z)$ for each!
- Solution: “lazy” $f(n)$
  - on input of length $n$, only run for $2n$ steps
  - if enough time to see should increase (over $f(n-1)$), do it; else, stay same
  - (alternate proof: give explicit $f(n)$ that grows slowly enough…)

Ladner’s Theorem

- Inductive definition of $f(n)$
  - $f(0) = 0$
  - $f(n)$: for $n$ steps compute $f(0)$, $f(1)$, $f(2)$,…

Ladner’s Theorem

- Key: $n$ eventually large enough to notice completed previous stage

Ladner’s Theorem

- $L = \{ x | x \in \text{SAT} \text{ if } f(|x|) \text{ even, } x \in \text{TRIV} \text{ if } f(|x|) \text{ odd } \}$

- $L \in \text{NP}$ since $f(|x|)$ can be computed in $O(n)$ time
A puzzle

- cover up nodes with c colors
- promise: never color “arrow” same as “blank”
- determine which kind of tree in poly(n, c) steps?

Introduction

- Ideas
  - depth-first-search; stop if see [arrow]
  - how many times may we see a given “arrow color”?
    - at most n+1
  - pruning rule?
    - if see a color > n+1 times, it can’t be an arrow node; prune
  - # nodes visited before know answer?
    - at most c(n+2)

Sparse languages and NP

- We often say NP-complete languages are “hard”

- More accurate: NP-complete languages are “expressive”
  - lots of languages reduce to them

Sparse languages and NP

- Sparse language: one that contains at most poly(n) strings of length ≤ n
- not very expressive – can we show this cannot be NP-complete (assuming P ≠ NP)?
  - [arrow]
  - yes: Mahaney 82 (homework problem)
- Unary language: subset of 1* (at most n strings of length ≤ n)
Sparse languages and \textbf{NP}

\textbf{Theorem} (Berman ’78): if a unary language is \textbf{NP}-complete then \textbf{P} = \textbf{NP}.

\begin{itemize}
  \item Proof:
    \begin{itemize}
      \item let \( U \subseteq 1^* \) be a unary language and assume \( \text{SAT} \leq U \) via reduction \( R \)
      \item \( \phi(x_1,x_2,\ldots,x_n) \) instance of \text{SAT}
    \end{itemize}
\end{itemize}

Sparse languages and \textbf{NP}

\begin{itemize}
  \item applying reduction \( R \):
    \begin{itemize}
      \item \( R(\phi(x_1,x_2,\ldots,x_n)) \)
      \item \( R(\phi(0,x_2,\ldots,x_n)) \)
      \item \( R(\phi(1,x_2,\ldots,x_n)) \)
      \item \( R(\phi(0,0,\ldots,0)) \)
      \item \( R(\phi(1,1,\ldots,1)) \)
    \end{itemize}
\end{itemize}

\begin{itemize}
  \item on input of length \( m = |\phi(x_1,x_2,\ldots,x_n)| \), \( R \) produces string of length \( \leq p(m) \)
  \item \( R \)'s different outputs are “colors”
    \begin{itemize}
      \item 1 color for strings not in \( 1^* \)
      \item at most \( p(m) \) other colors
    \end{itemize}
  \item puzzle solution \( \Rightarrow \) can solve \text{SAT} in \( \text{poly}(p(m)+1, n+1) = \text{poly}(m) \) time!
\end{itemize}

\textbf{Summary}

\begin{itemize}
  \item nondeterministic time classes:
    \begin{itemize}
      \item \textbf{NP}, \textbf{coNP}, \textbf{NEXP}
    \end{itemize}
  \item \textbf{NTIME} Hierarchy Theorem:
    \begin{itemize}
      \item \textbf{NP} \( \neq \) \textbf{NEXP}
    \end{itemize}
  \item major open questions:
    \begin{itemize}
      \item \textbf{P} \( \nsubseteq \) \textbf{NP}
      \item \textbf{NP} \( \nsubseteq \) \textbf{coNP}
    \end{itemize}
\end{itemize}
Summary

Remainder of lecture

- nondeterminism applied to space
- reachability
- two surprises:
  - Savitch’s Theorem
  - Immerman/Szelepcsényi Theorem