Last time:
- Finished NL = coNL
- Time and Space Hierarchy Theorems

Today:
- A “natural” intractable problem
- Oracles and P versus NP

Posted:
- Solutions to Problem Set 5
- Problem Set 6
Theorems:
SPACE\(o(f(n))\) \(\subsetneq\) SPACE\(f(n)\) for space constructible \(f\).
TIME\(o(f(n)/\log(f(n)))\) \(\subsetneq\) TIME\(f(n)\) for time constructible \(f\).

Corollary: NL \(\subsetneq\) PSPACE

Implies \(TQBF \notin\) NL because the polynomial-time reductions in the proof that \(TQBF\) is PSPACE-complete can be done in log space.

Check-in 22.1
Which of these are known to be true? Check all that apply.
(a) TIME\(2^n\) \(\subsetneq\) TIME\(2^{n+1}\)
(b) TIME\(2^n\) \(\subsetneq\) TIME\(2^{2n}\)
(c) NTIME\(n^2\) \(\subsetneq\) PSPACE
(d) NP \(\subsetneq\) PSPACE
**Defn:** $\text{EXPTIME} = \bigcup_k \text{TIME} \left( 2^{(n^k)} \right)$

$\text{EXPSPACE} = \bigcup_k \text{SPACE} \left( 2^{(n^k)} \right)$

Time Hierarchy Theorem

$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE}$

Space Hierarchy Theorem

**Defn:** $B$ is EXPTIME-complete if

1) $B \in \text{EXPTIME}$

2) For all $A \in \text{EXPTIME}$, $A \leq_P B$

Same for EXPSPACE-complete

**Theorem:** If $B$ is EXPTIME-complete then $B \notin P$

**Theorem:** If $B$ is EXPSPACE-complete then $B \notin \text{PSPACE}$ (and $B \notin P$)

Next will exhibit an EXPSPACE-complete problem
A “Natural” Intractable Problem

**Defn:** \( E_{Q_{REX}} = \{(R_1, R_2) | R_1 \text{ and } R_2 \text{ are equivalent regular expressions}\} 

**Theorem:** \( E_{Q_{REX}} \in \text{PSPACE} \)

**Proof:** Later (if time) or exercise (uses Savitch’s theorem).

**Notation:** If \( R \) is a regular expression write \( R^k \) to mean \( \underbrace{RR \cdots R}_k \) (exponent is written in binary).

**Defn:** \( E_{Q_{REX}^\uparrow} = \{(R_1, R_2) | R_1 \text{ and } R_2 \text{ are equivalent regular expressions with exponentiation}\} 

**Theorem:** \( E_{Q_{REX}^\uparrow} \) is \( \text{EXPSPACE}\)-complete

**Proof:**
1) \( E_{Q_{REX}^\uparrow} \in \text{EXPSPACE} \)
   2) If \( A \in \text{EXPSPACE} \) then \( A \leq_p E_{Q_{REX}^\uparrow} \)

1) Given regular expressions with exponentiation \( R_1 \) and \( R_2 \), expand the exponentiation by using repeated concatenation and then use \( E_{Q_{REX}} \in \text{PSPACE} \). The expansion is exponentially larger, so gives an \( \text{EXPSPACE} \) algorithm for \( E_{Q_{REX}^\uparrow} \).

2) Let \( A \in \text{EXPSPACE} \) be decided by TM \( M \) in space \( 2^{(n^k)} \).

**Give a polynomial-time reduction** \( f \) **mapping** \( A \) **to** \( E_{Q_{REX}^\uparrow} \).
Showing $A \leq_p E$Q$_{REX\uparrow}$

**Theorem:** $E$Q$_{REX\uparrow}$ is EXPSPACE-complete

Proof continued: Let $A \in$ EXPSPACE decided by TM $M$ in space $2^{(n^k)}$. Give a polynomial-time reduction $f$ mapping $A$ to $E$Q$_{REX\uparrow}$.

$$f(w) = \langle R_1, R_2 \rangle$$

$w \in A$ iff $L(R_1) = L(R_2)$

Construct $R_1$ so that $L(R_1) =$ all strings except a rejecting computation history for $M$ on $w$.

Construct $R_2 = \Delta^*$ ($\Delta$ is the alphabet for computation histories, i.e., $\Delta = \Gamma \cup Q \cup \{\#\}$)

**$R_1$ construction:** $R_1 = R_{bad-start} \cup R_{bad-move} \cup R_{bad-reject}$

Rejecting computation history for $M$ on $w$:

- $2^{(n^k)}$ - $2^{(n^k)}$ - $2^{(n^k)}$ - $2^{(n^k)}$

- $q_0w_1w_2 \ldots w_n \overleftarrow{\cdots} \overleftarrow{\#} \overleftarrow{ababa} \ldots \overleftarrow{abababa} \overleftarrow{\#} \ldots \overleftarrow{\#} \overleftarrow{\cdots} q_{reject} \overleftarrow{\cdots}$

$$C_1 = C_{start} \quad C_2 \quad C_{reject}$$

**Check-in 22.2**

Roughly estimate the size of the rejecting computation history for $M$ on $w$.

(a) $2^n$  
(b) $2^{(n^k)}$  
(c) $2^{2^{(n^k)}}$
Construct $R_1$ to generate all strings except a rejecting computation history for $M$ on $w$.

$R_1 = R_{\text{bad-start}} \cup R_{\text{bad-move}} \cup R_{\text{bad-reject}}$

Rejecting computation history for $M$ on $w$:

\[
\begin{array}{c}
\text{Construct } R_1 \text{ to generate all strings except a rejecting computation history for } M \text{ on } w.
\end{array}
\]

\[
R_{\text{bad-start}} \text{ generates all strings that do not start with } C_{\text{start}} = q_0 w_1 w_2 \cdots w_n \cdots
\]

\[
R_{\text{bad-start}} = S_0 \cup S_1 \cup S_2 \cup \cdots \cup S_n \cup S_{\text{blanks}} \cup S_{\#}
\]

Remember: $\Delta$ is the alphabet for computation histories, i.e., $\Delta = \Gamma \cup Q \cup \{\#\}$

Notation:

- $\Delta_{\epsilon} = \Delta \cup \{\epsilon\}$
- $\Delta_{-b} = \Delta$ without $b$
- $\Delta^7 = \text{all strings of length 7}$
- $\Delta_{\epsilon}^7 = \text{all strings of length 0 thru 7}$

\[
S_{\text{blanks}} = \Delta_{\epsilon}^{n+1} \Delta_{\epsilon}^{2^{(n^k)}-(n+2)} \Delta_{-\#} \Delta^* \text{ all strings of length } n + 1 \text{ thru } 2^{(n^k)} - 1
\]

\[
S_0 = \Delta_{-q_0} \Delta^*
\]

\[
S_1 = \Delta \Delta_{-w_1} \Delta^*
\]

\[
S_2 = \Delta^2 \Delta_{-w_2} \Delta^*
\]

\[
\vdots
\]

\[
S_{n+1} = \Delta^{n+1} \Delta_{-w_n} \Delta^*
\]

\[
S_{\#} = \Delta^2 \Delta_{-\#} \Delta^*
\]
\[ A \leq_P EQ_{\text{REX}} \uparrow (R_{\text{bad-move}} \& R_{\text{bad-reject}}) \]

Construct \( R_1 \) to generate all strings except a rejecting computation history for \( M \) on \( w \).

\[ R_1 = R_{\text{bad-start}} \cup R_{\text{bad-move}} \cup R_{\text{bad-reject}} \]

Rejecting computation history for \( M \) on \( w \):

- \( C_1 = C_{\text{start}} \)
- \( C_2 \)
- \( C_{\text{reject}} \)

\( R_{\text{bad-reject}} \) generates all strings that do not contain \( q_{\text{reject}} \)

\[ R_{\text{bad-reject}} = \Delta^* q_{\text{reject}} \]

\( R_{\text{bad-move}} \) generates all strings that contain an illegal \( 2 \times 3 \) neighborhood

\[ R_{\text{bad-move}} = \bigcup_{\text{illegal}} \left[ \Delta^* \text{abc} \Delta^{2(n^k)-2} \text{def} \Delta^* \right] \]

- \( C_i \)
- \( C_{i+1} \)
Let $A$ be any language.

**Defn:** A TM $M$ with oracle for $A$, written $M^A$, is a TM equipped with a “black box” that can answer queries “is $x \in A$?” for free.

**Example:** A TM with an oracle for $SAT$ can decide all $B \in NP$ in polynomial time.

**Defn:** $P^A = \{ B \mid B$ is decidable in polynomial time with an oracle for $A \}$

Thus $NP \subseteq P^{SAT}$

$NP = P^{SAT}$? Probably No because $coNP \subseteq P^{SAT}$

**Defn:** $NP^A = \{ B \mid B$ is decidable in nondeterministic polynomial time with an oracle for $A \}$

Recall $MIN\text{-}FORMULA = \{ \langle \phi \rangle \mid \phi$ is a minimal Boolean formula $\}$

**Example:** $MIN\text{-}FORMULA \in NP^{SAT}$

“On input $\langle \phi \rangle$:

1. Guess shorter formula $\psi$
2. Use $SAT$ oracle to solve the $coNP$ problem: $\phi$ and $\psi$ are equivalent
3. Accept if $\phi$ and $\psi$ are equivalent. Reject if not.”
Oracles and P versus NP

Theorem: There is an oracle $A$ where $P^A = NP^A$
Proof: Let $A = TQBF$
$NP^{TQBF} \subseteq NPSPACE = PSPACE \subseteq p^{TQBF}$

Relevance to the P versus NP question

Recall: We showed $EQ_{REXT} \notin PSPACE$.
Could we show $SAT \notin P$ using a similar method?

Reason: Suppose YES.
The Hierarchy Theorems are proved by a diagonalization.
In this diagonalization, the TM $D$ simulates some TM $M$.
If both TMs were oracle TMs $D^A$ and $M^A$ with the same oracle $A$,
the simulation and the diagonalization would still work.
Therefore, if we could prove $P \neq NP$ by a diagonalization,
we would also prove that $P^A \neq NP^A$ for every oracle $A$.
But that is false!

Check-in 22.3

Which of these are known to be true? Check all that apply.

(a) $p^{SAT} = \overline{p^{SAT}}$
(b) $NP^{SAT} = coNP^{SAT}$
(c) $MIN-FORMULA \in P^{TQBF}$
(d) $NP^{TQBF} = coNP^{TQBF}$
Quick review of today

1. Defined EXPTIME and EXPSPACE
2. Defined EXPTIME- and EXPSPACE-completeness
3. Showed $EQ_{\text{REX}^1}$ is EXPSPACE-complete and thus $EQ_{\text{REX}^1} \not\in \text{PSPACE}$
4. Defined oracle TMs
5. Showed $P^A = NP^A$ for some oracle $A$
6. Discussed relevance to the P vs NP question
Theorem: $EQ_{\text{REX}} \in \text{PSPACE}$

Proof: Show $EQ_{\text{REX}} \in \text{NPSPACE}$

“On input $\langle R_1, R_2 \rangle$ [assume alphabet $\Sigma$]

1. Convert $R_1$ and $R_2$ to equivalent NFAs $N_1$ and $N_2$ having $m_1$ and $m_2$ states.
2. Nondeterministically guess the symbols of a string $s$ of length $2^{m_1 + m_2}$ and simulate $N_1$ and $N_2$ on $s$, storing only the current sets of states of $N_1$ and $N_2$.
3. If they ever disagree on acceptance then accept.
4. If always agree on acceptance then reject.”