18.404/6.840 Recitation #12
Final Revision

Abbas Zeitoun
zeitoun@mit.edu
BPP

**Defn:** A probabilistic TM is a TM that can have either one or two choices at each step of computation.

If there are $k$ choices along branch $b$, the probability that we traverse $b$ is $Pr[\text{branch } b] = 2^{-k}$

$$Pr[ M \text{ accepts } w ] = \sum_{b \text{ accepts}} Pr[ \text{branch } b]$$

$$Pr[ M \text{ rejects } w ] = 1 - Pr[ M \text{ accepts } w ]$$

**Defn:** We say that a PTM $M$ decides language $A$ with error probability $\epsilon \geq 0$ if for every $w$, $Pr[ M \text{ gives the wrong answer about } w \in A ] \leq \epsilon$

**Defn:** $\text{BPP} = \{A| \text{ some poly-time PTM decides } A \text{ with error } \epsilon = \frac{1}{3} \}$
Interactive Proofs

**Defn:** An interactive proof system consists of:
1) Verifier (V): polynomial time PTM (has access to w)
2) Prover (P): unlimited computational power (has access to w)

P and V exchange a polynomial number of polynomial-size messages, and then V accepts or rejects.

\[ \text{IP} = \{ A \mid \text{for some V and some honest prover } P \]
\[ w \in A \rightarrow \Pr[ (V \leftrightarrow P) \text{ accepts } w ] \geq \frac{2}{3} \]
\[ w \notin A \rightarrow \text{for any prover } \tilde{P} \Pr[ (V \leftrightarrow \tilde{P}) \text{ accepts } w ] \leq \frac{1}{3} \]

\( \text{e.g. ISO, } \overline{ISO}, \#SAT \in \text{IP} \)
Important Theorems

**Savitch’s Theorem:** for any function \( f = \Omega(\log(n)) \):
\[
\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n)) \text{ (by reusing space)}
\]

**Space Hierarchy Theorem:** for any space-constructible function \( f \):
\[
\text{SPACE}\left(o(f(n))\right) \subsetneq \text{SPACE}(f(n))
\]

**Time Hierarchy Theorem:** for any time-constructible function \( f \):
\[
\text{TIME}\left(o\left(\frac{f(n)}{\log f(n)}\right)\right) \subsetneq \text{TIME}(f(n))
\]
Hierarchy of Time & Space Complexity Classes

Other useful relationships:
- NL \not\subseteq PSPACE \not\subseteq EXPSPACE
- P \not\subseteq EXPTIME
- REG \not\subseteq CFL \not\subseteq P
- REG \not\subseteq L
- P \subseteq BPP \subseteq PSPACE

To show proper subsets:
- For time complexity classes: Time Hierarchy
- For space complexity classes: Space Hierarchy
Proving Membership

- For all classes: provide a TM that decides the language while respecting the constraints of the class (time constraints, space constraints, non-determinism); reduce to a language in that class
- **L**: useful to use fixed number of pointers into the input, counters, simple arithmetic; useful to reuse space; log-space transducer to a language in L.
- **NL, coNL**: same as L, but since NL=coNL, can prove membership of the language or its complement.
- **NP**: can also show that a certificate is verifiable in polynomial time.
- **coNP**: usually easier to show the complement of the language is in NP.
- **PSPACE, NPSPACE**: reusing space, non-determinism.
- **EXP/EXPTIME**: can perform an exhaustive search
- **EXPSPACE**: can perform several exhaustive searches by reusing space.
Proving Non-Membership

- **To prove non-membership**: reduce from a complete language of a class known to be bigger than the class under consideration.
e.g. Reducing from TQBF proves that a language is not in L.

- **To state that membership is unlikely**: reduce from a complete language of a class we think is different from the class under consideration.
e.g. Reducing from SAT argues that a problem is not in P (because we suspect but don’t actually know if $P \neq NP$)

- **To prove existence of languages outside a class**: use hierarchy theorems, diagonalization.
Proving Completeness

- A language $B$ is C-complete for some class $C$ if:
  1) $B \in C$
  2) For all $A \in C$, $A \leq_p B$

- Exception: For NL-completeness, use log-space reducibility instead
Useful Proof Techniques

- **Gadgets:** During reductions, it’s usually useful to have gadgets for the important features of the objects in the language.  
  e.g. For Boolean formulas, have gadgets associated with variables and clauses. For graph problems, have gadgets associated with nodes and edges.

- **Computation Histories/Configuration Graphs:** also useful for reductions.

- **Arithmetization:** For some problems, it might be useful to convert a Boolean formula to an equivalent polynomial.

- **Amplification Lemma:** If some PTM decides a language $A$ with error $\epsilon_1 < \frac{1}{2}$, then there exists an equivalent PTM that decides $A$ with error $\epsilon_2 < \epsilon_1$ ($\epsilon_2$ can be as small as $2^{-\text{poly}(n)}$). Useful when bounding the error probability of a PTM.

- **Reusing Oracles:** If a TM has access to an oracle $A$, it can be useful to query the oracle multiple times to solve similar subproblems as part of solving the original problem.  
  e.g. Using a SAT oracle to find an explicit satisfying assignment to a Boolean formula.
Closure Properties

<table>
<thead>
<tr>
<th>Class</th>
<th>( \cup )</th>
<th>( \circ )</th>
<th>( \ast )</th>
<th>( \cap )</th>
<th>( \setminus )</th>
<th>( \cdot^R ) (reversal)</th>
<th>( \setminus )</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NL=coNL</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>P</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NP</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>Yes</td>
<td>?</td>
</tr>
<tr>
<td>coNP</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
<td>Yes</td>
<td>?</td>
</tr>
<tr>
<td>BPP</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IP=PSPACE=NPSPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EXP/EXPTIME</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EXPSPACE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Proof Ideas for Closure Properties

- **Union**: Construct a TM that simulates both TMs and accepts if either accepts.

- **Concatenation**: Try all ways of splitting the input string into two parts and simulate the TM on both parts, accepting if the TM accepts both.

- **Star**: Use non-determinism to guess where to split the input string. For deterministic classes, use dynamic programming. It’s an open problem whether $L$ is closed under star.

- **Intersection**: Construct a TM that simulates both TMs and accepts if both accept.

- **Complement**: Straightforward for deterministic classes and for NL=coNL. For NP and coNP, closure under complement would imply that NP=coNP, which is an open problem.

- **Reversal**: Read the input string in reverse but keep the rest of the logic of the TM as is.

- **Difference**: $A \setminus B = A \cap \overline{B}$. The results follow from the intersection and complement properties.