18.404 Recitation 12

Dec 4, 2020
Today’s Topics

● Review: Interactive Proof Systems (IP)
● Complexity Class Containments
● How to show $A$ is in Complexity Class
● How to show $A$ is Complexity Class Complete
● Review from 1st half of the semester
  ○ Closure Properties
  ○ Common Languages
  ○ Pumping Lemma for Regular Languages
  ○ Pumping Lemma for CFLs
Review: Interactive Proof Systems (IP)

Two Interacting Parties:

- Verifier (V) - Probabilistic Polynomial TM. Performs the *accepting/rejecting*
- Prover (P) - Unlimited computational power

\[ \text{IP} = \{ A \mid \text{for some } V \text{ and } P \text{ (where } P \text{ is honest)} \]
\[ w \in A \rightarrow \Pr[ (V \Leftrightarrow P) \text{ accepts } w ] \geq 2/3 \]
\[ w \notin A \rightarrow \Pr[ (V \Leftrightarrow \sim P) \text{ accepts } w ] \leq 1/3 \]
Review: Interactive Proof Systems (IP) (cont.)

\[ IP = \{ A \mid \text{for some } V \text{ and } P \text{ (where } P \text{ is honest)} \]

\[ w \in A \rightarrow \Pr[ (V \equiv P) \text{ accepts } w ] \geq 2/3 \]

\[ w \notin A \rightarrow \Pr[ (V \equiv \neg P) \text{ accepts } w ] \leq 1/3 \}

The \( \neg P \) is a malicious prover, trying to make \( V \) accept when it should not. In other words, if \( w \notin A \), then no prover exists that can make \( V \) accept it with probability, not even a malicious prover.

In other words: If \( w \in A \), **some** prover exists that makes \( V \) accept it with prob.

If \( w \notin A \), **no** prover may exist that makes \( V \) accept it with prob.
Complexity Class Containments

Fill lines (A—B) represent $A \subseteq B$, but $B \subseteq A$ is unknown.
Dotted lines (A---B) represent $A \subsetneq B$ (A is a strict subset of B).
Complexity Class Containments (cont.)

Main Takeaways:

- $L \subseteq NL$
- $NL \subseteq P$
- $P \subseteq NP$, $P \subseteq coNP$
- $coNP \subseteq P^{SAT}$, $NP \subseteq P^{SAT}$
- $NL \nsubseteq PSPACE$
- $PSPACE \nsubseteq EXPSPACE$
- $NP \nsubseteq EXPTIME$
- $NL = coNL$
- $IP = PSPACE$
- notSAT $\in P^{SAT}$
- SAT $\in P^{SAT}$
Complexity Class Containments (cont.)
# How to show $A$ is in Complexity Class

<table>
<thead>
<tr>
<th>Complexity Class</th>
<th>Description</th>
</tr>
</thead>
</table>
| $L$              | - Give deterministic log-space algorithm for $A$
|                  |  ○ Pointers to input and counters
|                  |  ○ Show $A \leq_L B$, for some $B \in L$
|                  |  Example: $\{<x,y> \mid x + x^R = y\}$
| $NL$             | - Give nondeterministic log-space algorithm for $A$
|                  | - Give nondeterministic log-space algorithm for $\neg A$ ($NL=coNL$)
|                  |  Examples: PATH, Strongly-Connected, Bipartiteness
| $NP$             | - Give nondeterministic poly-time algorithm for $A$
|                  | - Give a poly-length witness (certificate) and a deterministic poly-time checker
|                  |  Examples (all of which are NP-Complete too)
|                  |  SAT, 3SAT, CLIQUE, HAMPATH, Dominating-Set
| $coNP$           | - Show that $\neg A \in NP$
|                  |  Example: $EQ_{BP}$
### How to show $A$ is in Complexity Class (cont.)

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
</table>
| $\text{P}^{\text{SAT}}/\text{NP}^{\text{SAT}}$ | - Give deterministic poly-time algorithm for $A$ that can ask SAT questions  
  ○ use that SAT is NP-Complete, so that SAT-oracle can decide any language in NP  
  Example: SAT2 (PSet 6 #2) |
| PSPACE  | - Give deterministic/nondeterministic poly-space algorithm for $A$ or $\neg A$  
  ○ (PSPACE=coPSPACE=NPSPACE (Savitch))  
  ○ maybe recursive algorithm such that number of calls * space per call = polynomial  
  Examples: TQBF, GG |
## How to show $A$ is Complexity Class Complete

<table>
<thead>
<tr>
<th>Complexity Class</th>
<th>Requirements</th>
<th>Examples</th>
<th>Tips for 3SAT $\leq_p A$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL-Complete</td>
<td>Show that $A \in \text{NL}$&lt;br&gt; Show that $B \leq_L A$ or that $B \leq_L \neg A$ for some NL-Complete language $B$</td>
<td>PATH, Strongly-Connected</td>
<td>Design “gadgets” for variables and clauses&lt;br&gt; Relate between gadgets according to the given formula (e.g. connect a gadget for a literal to a gadgets of clauses containing it)&lt;br&gt; Enforce consistency for variables/literals (a variable may be assigned $true$ or $false$, but not both)</td>
</tr>
<tr>
<td>NP-Complete</td>
<td>Show that $A \in \text{NP}$&lt;br&gt; Show that $B \leq_p A$ for some NP-Complete language $B$</td>
<td>3SAT, CLIQUE, HAMPATH, Dominating-Set</td>
<td></td>
</tr>
</tbody>
</table>
How to show $A$ is Complexity Class Complete (cont.)

| PSPACE-Complete | Show that $A \in \text{PSPACE}$  
|                 | Show that $A$ is PSPACE-hard by showing one of the following:  
|                 | - $B \leq_{P} A$ for some NL-Complete language $B$ (e.g. TQBF $\leq_{P} \text{GG}$)  
|                 | - Use computation histories of PSPACE language (e.g. $\text{EQ}_{\text{NFA}}$ in PSet 6, #5)  
|                 | Examples: TQBF, GG, $\text{EQ}_{\text{NFA}}$ |
Material From 1st Half of Semester
## Review: Closure Properties

<table>
<thead>
<tr>
<th>Regular Languages</th>
<th>CFLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed</td>
<td>Closed</td>
</tr>
<tr>
<td>● Union</td>
<td>● Union</td>
</tr>
<tr>
<td>● Concatenation</td>
<td>● Concatenation</td>
</tr>
<tr>
<td>● Kleene Star</td>
<td>● Kleene Star</td>
</tr>
<tr>
<td>● Intersection</td>
<td>● Intersection</td>
</tr>
<tr>
<td>● Negation</td>
<td>● Negation</td>
</tr>
</tbody>
</table>

Not-Closed

| Intersection |
| Negation |

Note:

\[
\text{CFL} \cap \text{Reg. lang.} = \text{CFL}
\]
**Review: Common Languages**

<table>
<thead>
<tr>
<th>T-Decidable</th>
<th>T-Recognizable (undecidable)</th>
<th>T-coRecognizable (undecidable)</th>
<th>T-Unrecognizable (neither T-recog nor T-coRecog)</th>
</tr>
</thead>
<tbody>
<tr>
<td>● $A_{DFA}$</td>
<td>● $A_{TM}$</td>
<td>● $E_{TM}$</td>
<td>● $E_{TM}$</td>
</tr>
<tr>
<td>● $A_{NFA}$</td>
<td>● $\neg(E_{EQ_{CFG}})$</td>
<td>● $\neg(A_{TM})$</td>
<td>● $\neg(E_{EQ_{TM}})$</td>
</tr>
<tr>
<td>● $E_{DFA}$</td>
<td>● $HALT_{TM}$</td>
<td>● $E_{CFG}$</td>
<td></td>
</tr>
<tr>
<td>● $EQ_{DFA}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>● $A_{CFG}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>● $E_{CFG}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Review: Pumping Lemmas (Regular Language)

For every regular language, there exists a pumping number $p \geq 1$ such that every string of length at least $p$ can be written as $w=xyz$ and satisfies:

- $|y| \geq 1$
- $|xy| \leq p$
- $(\forall n \geq 0) (xy^n z \in L)$
Review: Pumping Lemmas (CFLs)

For every CFL, there exists a pumping number $p \geq 1$ such that every string of length at least $p$ can be written as $s=uvxyz$ and satisfies:

- $(\forall n \geq 0) \ (uv^nxy^nz \in L)$
- $|vy| \geq 1$
- $|vxy| \leq p$

Note: Need to make sure that all slide-windows in the string that CANNOT be pumped!