1 Complete Problems for \(PSPACE\)

1.1 Game-tree evaluation

Consider a game played by black and red. Black wins iff
\[ \exists \text{ move 1 for black} \quad \forall \text{ move 2 for red} \quad \exists \text{ move 3 for black} \quad \forall \text{ move 4 for red} \quad \cdots \quad \text{black wins}. \]

We can think of a universal game:

\( \text{TQBF (True Quantified Boolean Formulas): An instance is of the form } Q_1 x_1 Q_2 x_2 \cdots Q_\ell x_\ell \psi(x_1, \ldots, x_\ell) \]

where \(Q_i \in \{\exists, \forall\}\), and where \(x_i\)'s are boolean variables and \(\psi\) is a boolean formula. An instance is in \(\text{TQBF}\) if the statement is true.

Claim 1. \(\text{TQBF}\) is \(PSPACE\)-complete.

Proof. \(\text{TQBF} \in PSPACE\). The obvious algorithm to evaluate the formula runs in polynomial space when space is reused as much as possible.

Given a TM \(M\) which runs in space \(S(n)\) which is polynomial in \(n\). \(M\) has at most \(\exp(S(n))\) configurations, and \(M\) runs in time at most \(\exp(S(n))\). Just as we did last time, consider the configuration graph for \(M\) (where there is a vertex for each configuration, and there is an edge from a configuration \(c_1\) to a configuration \(c_2\) if \(M\) will transition from \(c_1\) to \(c_2\) in one step).

Consider the following game, where the first player wins if \(M\) accepts.

The first move for player 1 is to choose a configuration \(c_1\) such that after \(T = T(n)\) steps, \(M\) is in configuration \(c_1\). This is equivalent to saying that there exists a path from the start configuration to \(c_1\) of length \(T/2\), and that there exists a path from \(c_1\) to the accepting configuration of length \(T/2\).

Then player 2's move is to choose whether they don't believe the first part of the claim (there exists a path from the start configuration to \(c_1\) of length \(T/2\)), or the second part of the claim (that there exists a path from \(c_1\) to the accepting configuration of length \(T/2\)).

A “board position” can be viewed as a triple \((c_s, c_f, t)\). The first player wins if there is a path from \(c_s\) to \(c_f\) of length \(t\).

A move for the black player is to choose a configuration \(c_m\) such that there is a path of length \(t/2\) from \(c_s\) to \(c_m\) and one from \(c_m\) to \(c_f\) A move for the red player is \(b \in \{0, 1\}\). If \(b = 0\), then the new board position is \((c_s, c_m, t/2)\); if \(b = 1\), then the new board position is \((c_m, c_f, t/2)\).

This process is continued until \(t = 1\), in which case the black player wins if \(M\) goes from \(c_s\) to \(c_f\) in one step, and otherwise the red player wins.

The initial board position is \((c_{\text{start}}, c_{\text{accept}}, T(n))\). Since \(T(n) \leq \exp(S(n))\), the game will take at most \(\text{poly}(S(n))\) rounds.

If \(M\) accepts, then the black player has a winning strategy (playing honesty according to what \(M\) does).

If \(M\) rejects, then the red player can always choose \(b\) such that the recursive claim is false (if \(M\) reaches \(c_m\) from \(c_s\) in \(t/2\) steps, choose \(b = 1\), otherwise choose \(b = 0\).)

This game can be translated into a \(TQBF\) instance \(3c_1 \forall b_1 3c_2 \forall b_2 \cdots 3c_{(\log T)} \forall c_{(\log T)}\{ \text{figure out } c_s \text{ and } c_f \text{ corresponding to the board position and verify that } M \text{ goes from } c_s \text{ to } c_f \text{ in one step } \} \).
Recall that for each \( L \in \Sigma_i \) in the polynomial hierarchy, there exists \( V \in P \) such that \( x \in L \) if \( \exists y_1 \forall y_2 \cdots \exists y_i V(x, y_1, \ldots, y_i) \). We can think of the polynomial hierarchy as games with a fixed number of moves, whereas \( TQBF \), and therefore \( PSPACE \) correspond to games where the number of moves is polynomial in the size of the input. Thus \( P \subseteq NP, co-NP \subseteq PH \subseteq PSPACE \subseteq EXP \).

2 \( NL = co-NL \)

**Theorem 1** (Immerman-Szelepcsényi theorem). \( NSPACE(S(n)) = co-NSPACE(S(n)) \) for all \( S(n) \geq \log n \).

**Proof.** We will only prove that \( NL = co-NL \), larger space follows from a padding argument.

Recall that the problem of deciding whether there is a path from \( u \) to \( v \) in a graph \( G \) is \( NL \)-complete. We will show that there exists a graph \( G' \) and \( u' \) and \( v' \) such that if there is a path from \( u \) to \( v \) in \( G \), then there is not a path from \( u' \) to \( v' \) in \( G' \).

Idea: Inductive Counting. We will give a non-deterministic algorithm which does the following: Count the number of nodes reachable from \( u \) in \( \leq \ell \) steps. Use this to certify that nodes are not reachable in \( \leq \ell + 1 \) steps. Use this to count the number of nodes reachable from \( u \) in \( \leq \ell + 1 \) steps.

We’ll say a non-deterministic machine \( M \) computes a function \( f \) if for each run, either \( M \) rejects or accepts and outputs \( f(x) \). In addition, there exists a run where \( M \) accepts.

\( \text{Count}(u, \ell) \) will count the number of nodes reachable from \( u \) in at most \( \ell \) steps, and \( \text{Decide}(u, w, \ell) \) will decide whether there exists a path from \( u \) to \( w \) of length at most \( \ell \). These subroutines will both be non-deterministic and mutually recursive.

1. \( \text{Count}(u, \ell) \):
   1. \( \text{count} \leftarrow 0 \)
   2. For \( v \in V \) do:
      1. If \( \text{Decide}(u, v, \ell) \), \( \text{count} \leftarrow \text{count} + 1 \)
   5. Return \( \text{count} \)
2. \( \text{Decide}(u, w, \ell) \):
   1. \( \text{Count}(u, \ell - 1) \)
   3. \( \text{count}' \leftarrow 0 \)
   4. For \( x \in V \) do:
      1. Guess a path from \( u \) to \( x \) of length \( \ell - 1 \)
      6. If we reach \( x \), \( \text{count}' \leftarrow \text{count}' + 1 \)
      7. If we reach \( x \) and \( (x, w) \in E \), return “True”
   8. If \( \text{count}' \neq \text{count} \), REJECT
   9. If \( \text{count}' = \text{count} \), return “False”

If the algorithm doesn’t REJECT, then both subroutines compute the correct answer, and if the non-deterministic guesses are correct, then the algorithm doesn’t REJECT.

\( \text{Decide} \) uses space \( O(\log n) \): each of \( \text{count}, \text{count}' \), and \( x \) need \( \log n \) bits, and checking a path is a \( O(\log n) \) space procedure. \( \text{Count} \) uses space \( \log n \) to store \( \text{count} \). Together, they use space \( O(\log n) \). \( \square \)