PSPACE-Completeness

Def. A language \( B \) is PSPACE-complete if
1. it is in PSPACE
2. any \( A \) in PSPACE is polynomial time reducible to \( B \)

Just condition 2: \( \text{PSPACE-hard} \) (may be harder than PSPACE)

Reduction is still polynomial time (NOT space). Why?

Want reduction to be easy (\( A \) and \( B \) still qualitatively similar)

Increasing space by factor \( c \) might mean time increase by \( c^n \) !!

TQBF is PSPACE-complete

Easy part: in PSPACE.

if \( \phi = \forall x \psi \), evaluate \( \psi[x \leftarrow 0] \land \psi[x \leftarrow 1] \)
else if \( \phi = \exists x \psi \), evaluate \( \psi[x \leftarrow 0] \lor \psi[x \leftarrow 1] \)
else \( \phi \) must be constant expression, evaluate

Space: depth of stack, store one variable each \( \Rightarrow \) linear space

Quantified Boolean Formulas

We’ve seen quantifiers in predicate logic: \( \forall x \exists y P(x, y) \)

\( x, y, \ldots \): variables taking values from some universe

In general: infinite set of interpretations, can’t decide by (semantic) evaluation, need (syntactic) proofs.

Simpler: quantifiers over propositional formulas:

\( \forall x \exists y [(x \lor y) \land (x \lor y)] \) true
\( \exists x \forall y [(x \lor y) \land (x \lor y)] \) false

Can always bring to prenex normal form (all quantifiers in front)

Fully quantified formula (sentence, all vars bound): true or false

\( \text{TQBF} = \{ \langle \phi \rangle \mid \phi \text{ is a fully quantified Boolean formula} \} \)

Part 2: TQBF is PSPACE-hard

Let \( A \) be a language decided by \( M \) in space \( n^k \).

Map any string to QBF that is true if \( M \) accepts \( w \).

Space \( f(n) = n^k \Rightarrow \) no more than \( 2^d(n) \) configurations.

Construct formula \( \phi_{\text{start}, \text{accept}, h} \) meaning:

\( M \) can go from config \( c_1 \) to \( c_2 \) in at most \( t \) steps.

Top-level acceptance: \( \Phi_{\text{start}, \text{accept}, h} \), with \( h = 2^d(n) \)

Base case: \( t = 1 \): \( c_1 = c_2 \) or \( c_1 \rightarrow c_2 \) in one step of \( M \)

Can write boolean formula like in Cook-Levin theorem.

Recursion: split into \( \lfloor t/2 \rfloor \) and \( \lceil t/2 \rceil \): there must exist an intermediate configuration \( m_1 \)
Keeping the formula linear-size

First try: \( \phi_{c_1,c_2,t} = \exists m_1[\phi_{c_1,m_1,t/2} \land \phi_{m_1,c_2,t/2}] \)
Does not work: formula size doubles at each level.

Keeping it linear: factor out formula, use quantifiers
\( \phi_{c_1,c_2,t} = \exists m_1 \forall (c_3,c_4) \in \{(c_1,m_1),(m_1,c_2)\} \phi_{c_3,c_4,t/2} \)
Quantifier part added is linear in configuration size, so \( O(f(n)) \).
Number of levels: \( \log 2^{d(n)} \), thus \( O(f(n)) \)
Total formula size: \( O(f^2(n)) \), can be built in polynomial time

Winning Strategies for Games

Consider a QBF with alternating quantifiers.
\( \exists x_1 \forall x_2 \exists x_3 \ldots \psi \)
Players E and A alternate selecting values for the variable.
Player E wins if in the end, the formula is true.
Player A wins if in the end, the formula is false.

Does player E have a winning strategy?
This is exactly equivalent to \( \text{TQBF} \)!
(If formula is not alternating, players may make consecutive moves.
Can also insert quantifiers for dummy variables)

Geography Game on Directed Graph

Player who can’t move, loses. No repetitions allowed.
GG is in \( \text{PSPACE} \):

Geography Game is \( \text{PSPACE-hard} \): reduce from \( \text{TQBF} \)

Build game from formula.
(Assume \( \exists \) at start and end)
Structure forces alternation.
First pass determines value.
Tree structure forces win:
P2 chooses clause
P1 chooses literal
Can force side already played, other player loses.