QSAT is \textit{PSPACE}-complete

\textbf{Theorem:} QSAT is \textit{PSPACE}-complete.

\textbf{Proof:}

- in \textit{PSPACE}: \exists x_1 \forall x_2 \exists x_3 \ldots \text{Qx_n} \phi(x_1, x_2, \ldots, x_n)?
  - “exists”: for both \( x_1 = 0, x_1 = 1 \), recursively solve \forall x_2 \exists x_3 \ldots \text{Qx_n} \phi(x_1, x_2, \ldots, x_n)?
    - if at least one “yes”, return “yes”; else return “no”
  - “forall”: for both \( x_1 = 0, x_1 = 1 \), recursively solve \exists x_2 \forall x_3 \ldots \text{Qx_n} \phi(x_1, x_2, \ldots, x_n)?
    - if at least one “no”, return “no”; else return “yes”

- base case: evaluating a 3-CNF expression
  - poly(n) recursion depth
  - poly(n) bits of state at each level

QSAT is \textit{PSPACE}-complete

- given TM \( M \) deciding \( L \in \text{PSPACE} \); input \( x \)
  - \( 2^n \) possible configurations
  - single \text{START} configuration
  - assume single \text{ACCEPT} configuration

- define:
  \( \text{REACH}(X, Y, i) \iff \text{configuration } Y \text{ reachable from configuration } X \text{ in at most } 2^i \text{ steps.} \)

- Goal: produce 3-CNF \( \psi(w_1, w_2, \ldots, w_m) \) such that
  \[ \exists w_1 \forall w_2 \ldots \exists w_m \phi(w_1, \ldots, W) \iff \text{REACH(START, ACCEPT, n)} \]

- for \( i = 0, 1, \ldots, n^k \) produce quantified Boolean expressions \( \psi(A, B, W) \) such that \( \forall A, B: \)
  \[ \exists w_1 \forall w_2 \ldots \exists V \phi(A, B, W) \iff \text{REACH(A, B, i)} \]

- convert \( \psi_* \) to 3-CNF \( \phi \)
  - add \( V \)

- hardwire \( A = \text{START}, B = \text{ACCEPT} \)
  \[ \exists w_1 \forall w_2 \ldots \exists V \phi(W, V) \iff x \in L \]
QSAT is PSPACE-complete

Key observation:
\[
\text{REACH}(A, B, i+1) \iff \exists Z \[ \text{REACH}(A, Z, i) \land \text{REACH}(Z, B, i) \]
\]

- cannot define \( \psi_{i+1}(A; B; Z, W, W') \) to be
  \[
  \exists Z \[ \exists w_1 \forall w_2 \ldots \psi_i(A, Z, W) \land \exists w_1' \forall w_2' \ldots \psi_i(Z, B, W') \]
  (why?)

- define \( \psi_{i+1}(A; B; Z, X, Y, W) \) to be
  \[
  \exists Z \forall X \forall Y \[ (X = A \land Y = Z) \lor (X = Z \land Y = B) \implies \exists w_1 \forall w_2 \ldots \psi_i(X, Y, W) \]
  (\( \psi_0(A, B) = \text{true} \iff A = B \text{ or } A \text{ yields } B \text{ in 1 step} \))

- total size of \( \psi_n \) is \( O(n^k) \)

PSPACE and games

\( \psi_i(A, B) = \text{true iff } A = B \text{ or } A \text{ yields } B \text{ in 1 step} \)
\[
\psi_i(A; B; Z, X, Y, W) = \exists Z \forall X \forall Y \[(X = A \land Y = Z) \lor (X = Z \land Y = B) \implies \exists w_1 \forall w_2 \ldots \psi_i(X, Y, W)\]
\]

- \( |\psi_0| = O(n^k) \)
- \( |\psi_{i+1}| = O(n^k) + |\psi_i| \)
- reduction runs in polynomial time

PSPACE

Theorem: GEOGRAPHY is PSPACE-complete.

Proof:
- in PSPACE (proof?)
- PSPACE-hard. reduction from QSAT.

QSAT = \{ \( \varphi : \varphi \) is a 3-CNF, and \( 3x_1 \land x_2 \land x_3 \land \ldots \land x_n \implies \varphi(x_1, x_2, x_3, \ldots, x_n) \) \}

- Think of as 2-player game (player 1 trying to satisfy \( \varphi \); player 2 adversary):
  - player 1 picks truth value for \( x_1 \)
  - player 2 picks truth value for \( x_2 \)
  - player 1 picks truth value for \( x_i \)...
  - \( \varphi \in \text{QSAT} \) iff player 1 can win no matter what player 2 does.

GEOGRAPHY = \{ (G, s) : G is a directed graph and player I can win from node s \}

General phenomenon: many 2-player games are PSPACE-complete.
- 2 players I, II
- alternate picking edges
- lose when no unvisited choice

PSPACE

Theorem: GEOGRAPHY is PSPACE-complete.

Proof:
- in PSPACE (proof?)
- PSPACE-hard. reduction from QSAT.
GEOGRAPHY is PSPACE-complete

- We are reducing from the language:

\[ \text{QSAT} = \{ \phi : \phi \text{ is a 3-CNF, and} \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n \phi(x_1, x_2, x_3, \ldots, x_n) \} \]

...to the language:

\[ \text{GEOGRAPHY} = \{(G, s) : G \text{ is a directed graph and player I can win from node } s\} \]

Outline

- Challenges to Extended Church-Turing
  - randomized computation
  - quantum computation

Extended Church-Turing Thesis

- The "extended" Church-Turing Thesis:
  - everything we can compute in time \( t(n) \)
  - on a physical computer can be computed on a Turing Machine in time \( t(n)^{O(1)} \) (polynomial slowdown)
  - randomized computation challenges this belief

Randomness in computation

- Example of the power of randomness
- Randomized complexity classes

Outline
Communication complexity

- Goal: compute \( f(x, y) \) while communicating as few bits as possible between Alice and Bob
- Count number of bits exchanged (computation free)
- At each step: one party sends bits that are a function of held input and received bits so far

Communication complexity

- Simple function (equality):
  \[
  \text{EQ}(x, y) = 1 \text{ iff } x = y
  \]
- Simple protocol:
  - Alice sends \( x \) to Bob (\( n \) bits)
  - Bob sends \( \text{EQ}(x, y) \) to Alice (1 bit)
  - Total: \( n + 1 \) bits
  - (Works for any predicate \( f \))

Communication complexity

- Can we do better?
  - Deterministic protocol?
  - Probabilistic protocol?
    - At each step: one party sends bits that are a function of held input and received bits so far and the result of some coin tosses
    - Required to output \( f(x, y) \) with high probability over all coin tosses

Theorem: No deterministic protocol can compute \( \text{EQ}(x, y) \) while exchanging fewer than \( n + 1 \) bits.

Proof:
- "Input matrix":

\[ X = \{0,1\}^n \quad Y = \{0,1\}^n \]

\[ f(x, y) \]

- A sends 1 bit depending only on \( x \):
  - \( X = \{0,1\}^n \quad Y = \{0,1\}^n \)
  - Inputs \( x \) causing A to send 1
  - Inputs \( x \) causing A to send 0

- B sends 1 bit depending only on \( y \) and received bit:
  - \( X = \{0,1\}^n \quad Y = \{0,1\}^n \)
  - Inputs \( y \) causing B to send 1
  - Inputs \( y \) causing B to send 0
Communication complexity

- at end of protocol involving k bits of communication, matrix is partitioned into at most $2^k$ combinatorial rectangles
- bits sent in protocol are the same for every input $(x, y)$ in given rectangle
- conclude: $f(x,y)$ must be constant on each rectangle

- any partition into combinatorial rectangles with constant $f(x,y)$ must have at least $2^n + 1$ rectangles
- protocol that exchanges $\leq n$ bits can only create $2^n$ rectangles, so must exchange at least $n+1$ bits.

Matrix for EQ:

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0
\end{array}
\]