POLYNOMIAL SPACE

- QSAT: decide whether $\exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n \phi$ holds.
  where ϕ a Boolean expression in CNF with variables $x_1, \ldots, x_n$.
- QSAT is **PSPACE**-complete.
- Recall $\mathbf{AP} = \mathbf{ATIME}(n^k)$ is a class of languages decided in polynomial time by *alternating* Turing machines.
- QSAT is **AP**-complete.
  Hence, $\mathbf{AP} = \mathbf{PSPACE}$
- Many other **PSPACE**-complete problems:
  games, decision making, interactive proofs, verification, periodic optimization, . . .

Games

- QSAT is an example of a *two-person game*:
  - two players: $\exists$ and $\forall$
  - players move alternatingly ($\exists$ first)
  - a move: determining the truth value of a variable
  - $\exists$ tries to make the formula ϕ true and $\forall$ false.
  - after n moves either $\exists$ or $\forall$ wins.

Games—cont'd

- A game
  - two players move alternatingly on a “board”.
  - the number of moves is bounded by a polynomial in the size of the board
  - In the end some positions are considered a win of one player and the rest of the other.
- Solution: a winning strategy (an exponential object).
- Examples: chess, checkers, Go, nim, tic-tac-toe . . .

Games—cont'd

- How to separate computationally hard games from easy ones?
- Complexity theory cannot be used directly because games are played typically on a *fixed size* board.
- A possible solution: generalize the game to an arbitrary size board.
- GEOGRAPHY game:
  - Two players: I and II and board G (graph).
  - Move: select an unvisited neighbor of the current node.
  - The first player that cannot continue loses.

Games—cont'd

- GEOGRAPHY: Given a graph G and a starting node 1, is it a win for I.
- GEOGRAPHY is **PSPACE**-complete.
- GO is **PSPACE**-complete

Decision-making under uncertainty

- Repeatedly a decision followed by a random event
- SSAT (Stochastic satisfiability):

$$\exists x_1 \mathrm{R} x_2 \exists x_3 \mathrm{R} x_4 \cdots \mathbf{prob}[\phi(x_1, \ldots, x_n) = \mathbf{True}] > 1/2$$

- **APP**: languages decided by probabilistic alternating polynomial-time Turing machines.

Decision-making—cont'd

- A probabilistic alternating polynomial-time Turing machine:
  - a precise nondeterministic machine with uniformly two nondeterministic choices alternating between states in two disjoint sets K_+ and K_{MAX}.
  - M accepts x if for each K_{MAX} state there is a choice of one of the two successors such that if we consider the resulting computation tree, a majority of the leaves is accepting.
- **APP**: languages decidable by APP machines.
- $\mathbf{APP} = \mathbf{PSPACE}$
- SSAT is **PSPACE**-complete

Interactive Protocols

- Interactive proof systems closely related to *bounded* probabilistic alternating polynomial-time Turing machines (**ABPP**).
- ABPP machine:
  - if $x \in L$, then for each K_{MAX} state there is a choice of one of the two successors such that if we consider the resulting computation tree, 3/4 of the leaves are accepting and
  - if $x \notin L$, then for all choices of successors in K_{MAX} states at most 1/4 of the leaves are accepting in the resulting tree.
- $\mathbf{ABPP} \subseteq \mathbf{IP} \subseteq \mathbf{APP}$
- Shamir's theorem: $\mathbf{IP} = \mathbf{PSPACE}$
- $\mathbf{ABPP} = \mathbf{APP} = \mathbf{IP} = \mathbf{PSPACE}$

Verification

- General questions about Turing machines are undecidable
- IN-PLACE ACCEPTANCE: given a deterministic TM M and an input x, does M accept x *without ever leaving the $|x|+1$ first symbols of its string*?
- IN-PLACE ACCEPTANCE is **PSPACE**-complete
- IN-PLACE DIVERGENCE: given the description M of a deterministic TM, does M has a divergent computation that uses at most $|M|$ symbols?
- IN-PLACE DIVERGENCE is **PSPACE**-complete

Distributed Systems

- A system of communicating processes:
  $((V_1, E_1), \ldots, (V_n, E_n), P)$
  where (V_i, E_i) is a directed graph and P is a set of communication pairs $\{e_j, e'_j\}$ with $e_j \in E_k, e'_j \in E_l, k \neq l$.
- A system state $s \in V = V_1 \times \cdots \times V_n$
- Transition relation $T \subseteq V \times V$ of the system:
  $((a_1, \ldots, a_n), (b_1, \ldots, b_n)) \in T$ iff there are k, l such that $k \neq l, \{(a_k, b_k), (a_l, b_l)\} \in P, a_i = b_i$ for all $i \notin \{k, l\}$.

Verification of Distributed Systems

- Deadlock: a system state s with no successors (no s' such that $(s, s') \in T$)
- Determining whether a system has a deadlock system state is **NP**-complete
- Determining whether a system has a *reachable* (in T) deadlock system state is **PSPACE**-complete

Periodic Optimization

- PERIODIC SAT is **PSPACE**-complete

$$(x \vee \neg y_{+1}) \wedge (x \vee y)$$

- PERIODIC GRAPH COLORING is **PSPACE**-complete