POLYNOMIAL SPACE

Polynomial space-bounded computation has a variety of alternative characterizations and natural complete problems.

- QSAT
- Games
- Verification
- Periodic Optimization

(C. Papadimitriou: *Computational Complexity*, Chapter 19)

QSAT

- QSAT (QBF): given a Boolean expression ϕ in CNF with variables $x_1, \ldots, x_n$, is there is a truth value for the variable x_1 such that for both truth values of x_2 there is a truth value for x_3 and so on up to x_n, such that ϕ is satisfied by the overall truth assignment?

$$\exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n \phi$$

- QSAT is a generalization of the $\Sigma_i\mathbf{P}$-complete problem QSAT$_i$.
- QSAT is **PSPACE**-complete.

Polynomial space—cont'd

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

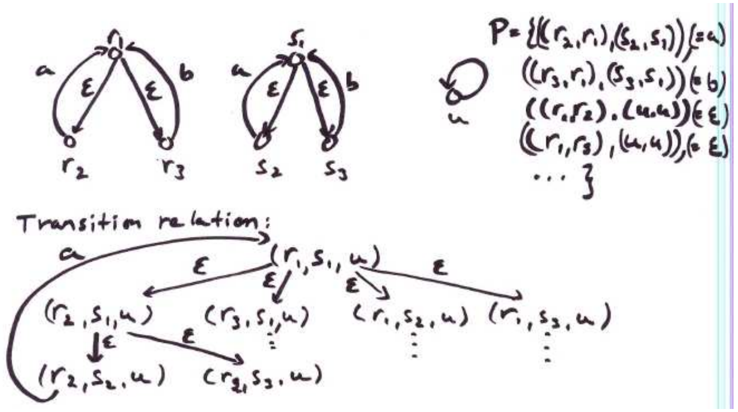
Verification

- General questions about Turing machines are undecidable.
- Restrictions lead to decidable but often computationally challenging problems (**PSPACE**-hard).
- 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 have 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 $i \notin \{k,l\}$.

Example. Consider a distributed system $((V_r,E_r),(V_s,E_s),(V_u,E_u),P)$



Verification of Distributed Systems

- Checking whether a design satisfies given specifications is often computationally challenging (**PSPACE**-hard).
- Deadlock: a system state s with no successors (no s' such that $(s,s') \in T$).

  Example. In the previous example the system state (r_2,s_3,u) is a deadlock.
- Determining whether a system has a deadlock system state is **NP**-complete.
- Given a system and an initial state, determining whether the system has a deadlock system state *reachable* (in T) from the initial state is **PSPACE**-complete.

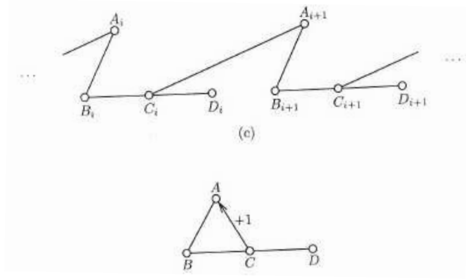
  Example. In the previous example the deadlock (r_2,s_3,u) is reachable from the system state (r_1,s_1,u).

Periodic Optimization

- What happens if the input is periodic (infinite in both directions)?

  $$\cdots \wedge \quad (x_i \vee \neg y_{i+1}) \wedge (x_i \vee y_i)$$
  $$\wedge \quad (x_{i+1} \vee \neg y_{i+2}) \wedge (x_{i+1} \vee y_{i+1}) \quad \wedge \cdots$$

  which can be written succinctly as $(x \vee \neg y_{+1}) \wedge (x \vee y)$.
- PERIODIC SAT is **PSPACE**-complete
- PERIODIC GRAPH COLORING is **PSPACE**-complete



Notice that here a legal coloring is possible with *2 colors*.

[Papadimitriou, 1994]