

More about PSPACE

With respect to  $\leq_P$

Thm: QBF is PSPACE-complete.

↑ the language of all true quantified Boolean formulas (all variables bound by quantifiers)

Proof: ① QBF  $\in$  PSPACE by the following recursive algorithm:

- QBF( $\forall x \varphi(x)$ ) = QBF( $\varphi(0)$ ) & QBF( $\varphi(1)$ )
  - QBF( $\exists x \varphi(x)$ ) = QBF( $\varphi(0)$ )  $\vee$  QBF( $\varphi(1)$ )
  - QBF( $\varphi \vee \psi$ ) = QBF( $\varphi$ )  $\vee$  QBF( $\psi$ )
  - QBF( $\varphi \& \psi$ ) = QBF( $\varphi$ ) & QBF( $\psi$ )
  - QBF( $\neg \varphi$ ) =  $\neg$ QBF( $\varphi$ )
- }  $O(n)$  levels of recursion, }  $O(n^2)$  space  
 }  $O(n)$  space per level. }

② QBF is PSPACE-hard: consider  $L \in$  PSPACE, TM  $M$  deciding  $L$  and its config. graph  $G$ .

- vectors of variables  $\bar{x}$  encoding vertices ...  $O(\text{poly}(n))$  bits [log of  $|G|$ ]
- formula  $\varphi(\bar{x}, \bar{y}) \equiv (\bar{x} = \bar{y}) \vee (x, y) \in E(G)$   
 - can construct poly-sized circuit as in proof of Cook-Levin thm.  
 & then reduce the circuit to an existentially-quantified formula as in Circuit-SAT  $\leq_P$  SAT.

• mimic proof of Savitch's thm.

Failed attempt:  $\varphi_k(\bar{x}, \bar{y}) \equiv \exists \bar{z} (\varphi_{k-1}(\bar{x}, \bar{z}) \& \varphi_{k-1}(\bar{z}, \bar{y}))$

Double recursion  $\Rightarrow$  formula size grows exponentially!

Better:  $\varphi_k(\bar{x}, \bar{y}) \equiv \exists \bar{z} \forall \bar{a} \forall \bar{b} ((\bar{a} = \bar{x} \& \bar{b} = \bar{z}) \vee (\bar{a} = \bar{z} \& \bar{b} = \bar{y})) \Rightarrow \varphi_{k-1}(\bar{a}, \bar{b})$

•  $G$  has size  $O(2^{\text{poly}(n)}) \Rightarrow \log(\text{path len}) \in \text{poly}(n) \Rightarrow$  recursion has  $\text{poly}(n)$  levels, formula size grows to  $\text{poly}(n)$ .

Intuition: PSPACE is the class of strategies for 2-player games with perfect information:

( $\exists$  player 1 move) ( $\forall$  player 2's response) ( $\exists$  player 1's counter-response) ( $\forall \dots$ ) ...

Examples:

- graph coloring game: undirected graph, finite set of  $k$  colors each player colors an uncolored vertex, every no edge must have both ends of the same color
- graph path game: building path edge by edge, ~~first~~ player has his target, ~~2nd player~~ vertices must not repeat
- variants of Go, checkers &c. (generalized to  $M \times N$  boards)
- Sokoban (1-player, but enough internal state, which restricts future moves, but can be modified)

these are known to be PSPACE-complete

Alternating Turing Machine (ATM)

- 3 kinds of states:
- deterministic: config is accepting  $\Leftrightarrow$  next config is accepting
  - existential: ~~accepts~~ <sup>config is accepting</sup>  $\Leftrightarrow \exists$  non-det. choice ~~leading to~~ <sup>leading to</sup> accepting config.
  - universal: is accepting  $\Leftrightarrow \forall$  non-det. choice leads to accepting config.
- We will require all computations to halt.

ATM  $\rightarrow$  classes  $ATIME(f), PSPACE, AP \dots$  [we won't define space-bounded classes as we require all computations to halt & alarm clocks don't help]

Theorem:  $AP = PSPACE$ .

Proof:  $\supseteq$  is easy:  $QBF \in AP$  since we can execute quantifiers using corresponding state types. So if  $L \leq_m QBF$ , we can first compute the reduction and then solve  $QBF$ .

- $\subseteq$ : ~~essentially~~ simulate the ATM recursively as in ~~QBF~~  $QBF \in PSPACE$ .
  - configurations take  $O(\text{poly}(n))$  space - all computations are poly-time, so they are poly-space, too.
  - recursion depth is bounded by time of the ATM.

Polynomial Hierarchy

Consider following restrictions of QBF:

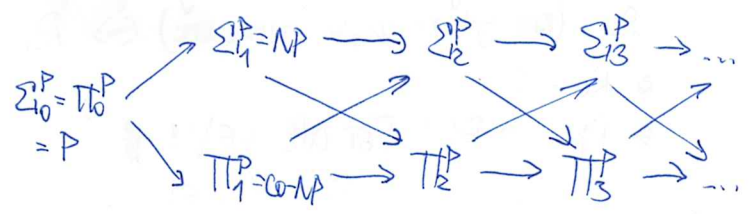
- $\Sigma_k$ -formulas:  $(\exists x_1 \dots \exists x_k)(\forall \dots)(\exists \dots) \dots \psi(\dots)$ 
  - $\underbrace{\quad}_{k \text{ groups of quantifiers, starting with } \exists}$
  - $\underbrace{\quad}_{\text{quantifier-free formula}}$
- $\Pi_k$ -formulas: similar, but starting with  $(\forall \dots)$
- $\Sigma_k$ -SAT :=  $\{ \langle \psi \rangle \mid \psi \text{ is a true } \Sigma_k\text{-formula} \}$  ... similarly  $\Pi_k$ -SAT.
- $\Sigma_1$ -SAT is SAT (for general formulas, not only CNF),  $\Pi_1$ -SAT is TAUT.

negation of a  $\Sigma_k$ -formula can be written as a  $\Pi_k$ -formula & vice versa

We can use this to define new classes which generalize NP:

- $\Sigma_k^P := \{ L \mid L \leq_m^P \Sigma_k\text{-SAT} \}$  ...  $\Sigma_1^P = NP, \Sigma_0^P = P$
- $\Pi_k^P := \{ L \mid L \leq_m^P \Pi_k\text{-SAT} \}$  ...  $\Pi_1^P = \text{co-NP}, \Pi_0^P = P, \Pi_k^P = \text{co-}\Sigma_k^P$
- generally, the following inclusions hold:

we defined classes using a problem complete for them



This is akin to the arithmetical hierarchy, but the inclusions are not known to be strict.

$\bullet PH := \bigcup_k \Sigma_k^P = \bigcup_k \Pi_k^P$

$\bullet$  since every  $\Sigma_k/\Pi_k$ -SAT reduces trivially to QBF, we have  $PH \subseteq PSPACE$  (not known to be strict)

Example: "given Bool. formula  $\psi$ , find the shortest  $\varphi$  s.t.  $\forall \bar{x} \varphi(\bar{x}) \Leftrightarrow \psi(\bar{x})" \in \Sigma_2^P$  (in fact, it's  $\Sigma_2^P$ -complete)

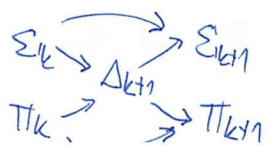
Remark: Definition using oracle machines also works:

$\Sigma_{k+1}^P := NP[\Sigma_k^P] = NP[\Pi_k^P]$

$\Pi_{k+1}^P := \text{co-NP}[\Sigma_k^P] = \text{co-NP}[\Pi_k^P]$

we can add

$\Delta_{k+1}^P := P[\Sigma_k^P] = P[\Pi_k^P]$



this leads to the same hierarchy

Remark: We can also define  $\Sigma_k^P$  &  $\Pi_k^P$  using alternative TMs:

- $\Sigma_k\text{-TIME}(f) := \{ L \mid L \text{ can be decided by an ATM running in time } \leq f(\text{input}) \text{ which performs at most } k-1 \text{ quantifier changes, starting with } \exists \}$
- $\Sigma_k^P = \Sigma_k\text{-TIME}(poly(n))$

Collapse of PH

- $P = NP \Leftrightarrow P = PH$
- $NP = co-NP \Leftrightarrow NP = PH$
- if  $\Sigma_j^P = \Sigma_{j+1}^P$ , then  $\Sigma_j^P = \Sigma_k^P$  for all  $k > j$  ] we say that PH collapsed to the j-th level
- ... so  $PH = \Sigma_j^P$
- if  $\Sigma_j^P = \Pi_j^P$ , then  $\Sigma_{j+1}^P = \Sigma_j^P$  (we can reduce  $\exists x \forall y \psi(x, y)$  to  $\exists x \exists y \psi(x, y)$ )

this is weaker than  $P = NP$ , but still open

Note: If graph isomorphism is in P, then  $PH = \Sigma_2^P$ . [proof non-trivial]

SPACE CO-CLASSES

Unlike non-deterministic time classes, non-det. space classes are known to be closed under complement.

Theorem (Immerman-Szelepcsényi):  $NSPACE(s(n)) = co-NSPACE(s(n))$  for all space-constructible functions  $s(n) \geq \log n$ .

Proof: We design a non-deterministic algorithm for non-reachability in config. graphs. More generally, we'll calculate  $R_i := \{\text{vertices reachable from source by walk of len } \leq i\}$ . Then modify graph by adding edges from target to all vertices, so  $(\text{target reachable from src}) \Leftrightarrow R_n = n$  for  $n = \#\text{vertices}$ .

$V_i := \text{set of these vertices}$

- $R_0 = 1$
- $R_{i-1} \rightarrow R_i$ : For all  $v \in V$ :  
For all  $w \in V_{i-1}$ :  
if  $(w, v) \in E$  or  $v = w$ :  $R_i \leftarrow R_{i-1} \cup v$

Enumeration of  $V_i$ :  
 $t \leftarrow 0$   
 For all  $u \in V$ :  
 If guess that  $u \in V_i$ :  
 If  $\nexists$  walk  $\text{src} \rightarrow u$  of length  $\leq i$ : REJECT  
 $t \leftarrow t + 1$   
 If  $t \neq R_{i-1}$ : REJECT

if we don't guess correctly, either the path doesn't exist or  $t < R_{i-1}$  at the end

guess the path using non-determinism & check that it's valid

Space needed:  $\bullet O(1)$  variables for vertices & counters }  $O(\log |V|)$  space, where  $|V| = 2^{O(s(n))}$  } so this is in  $NSPACE(s(n))$ .  
 $\bullet R_i$  and  $R_{i-1}$