Commutative Queries

Richard Beigel

Yale University &
University of Maryland College Park

Richard Chang

University of Maryland Baltimore County

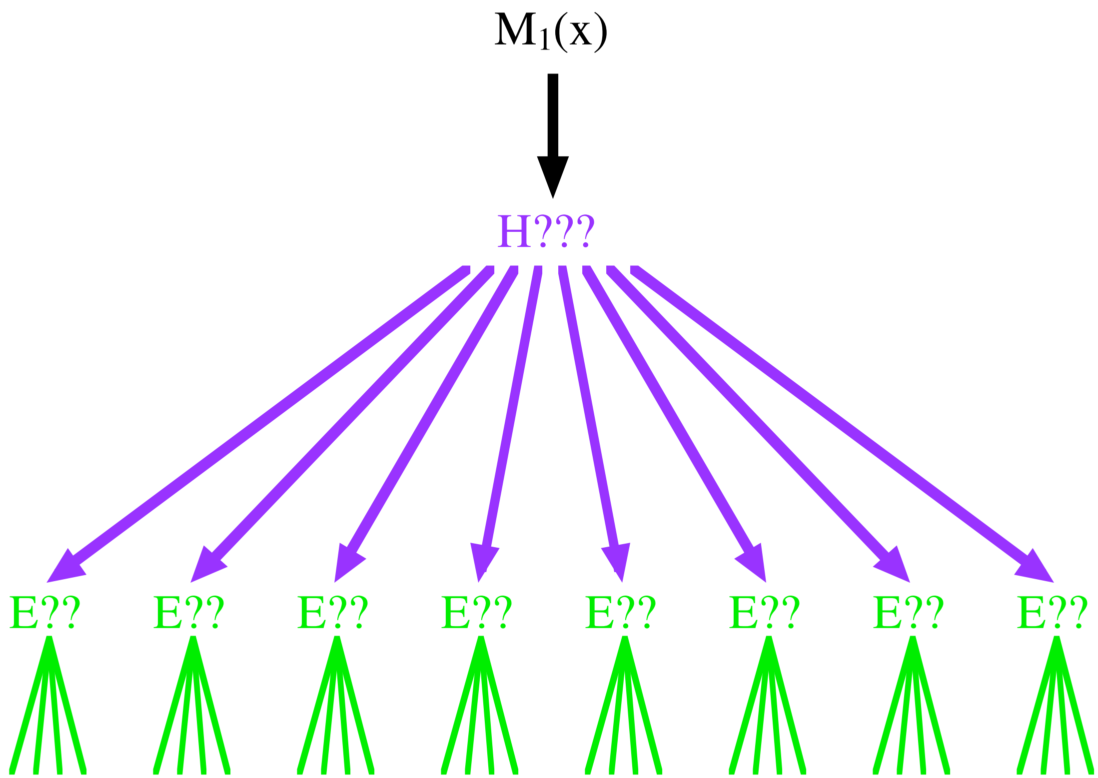
order of access

Given access to two oracles, which oracle should be queried first?
Does it matter?

- oracles must have different complexity
- complete languages of the Polynomial Hierarchy Σ_j^{P} and Σ_k^{P}, where $j < k$
- allow truth-table queries to each oracle
- recognize languages or compute functions

$\mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}} = r$ truth-table queries to H followed by s truth-table queries to E

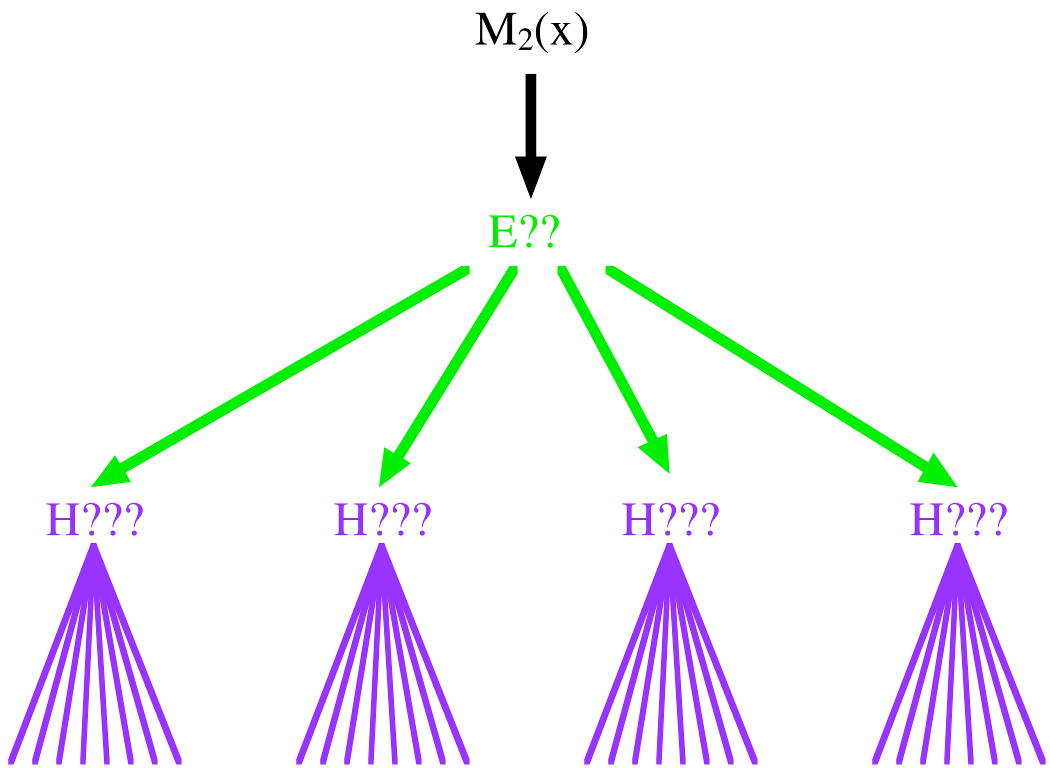
$E =$ easy oracle, Σ_j^{P} complete $\qquad H =$ hard oracle, Σ_k^{P} complete $\qquad j < k$



asking easy questions first

$\mathrm{P}^{E_{s\text{-tt}};H_{r\text{-tt}}} = s$ truth-table queries to E followed by r truth-table queries to H

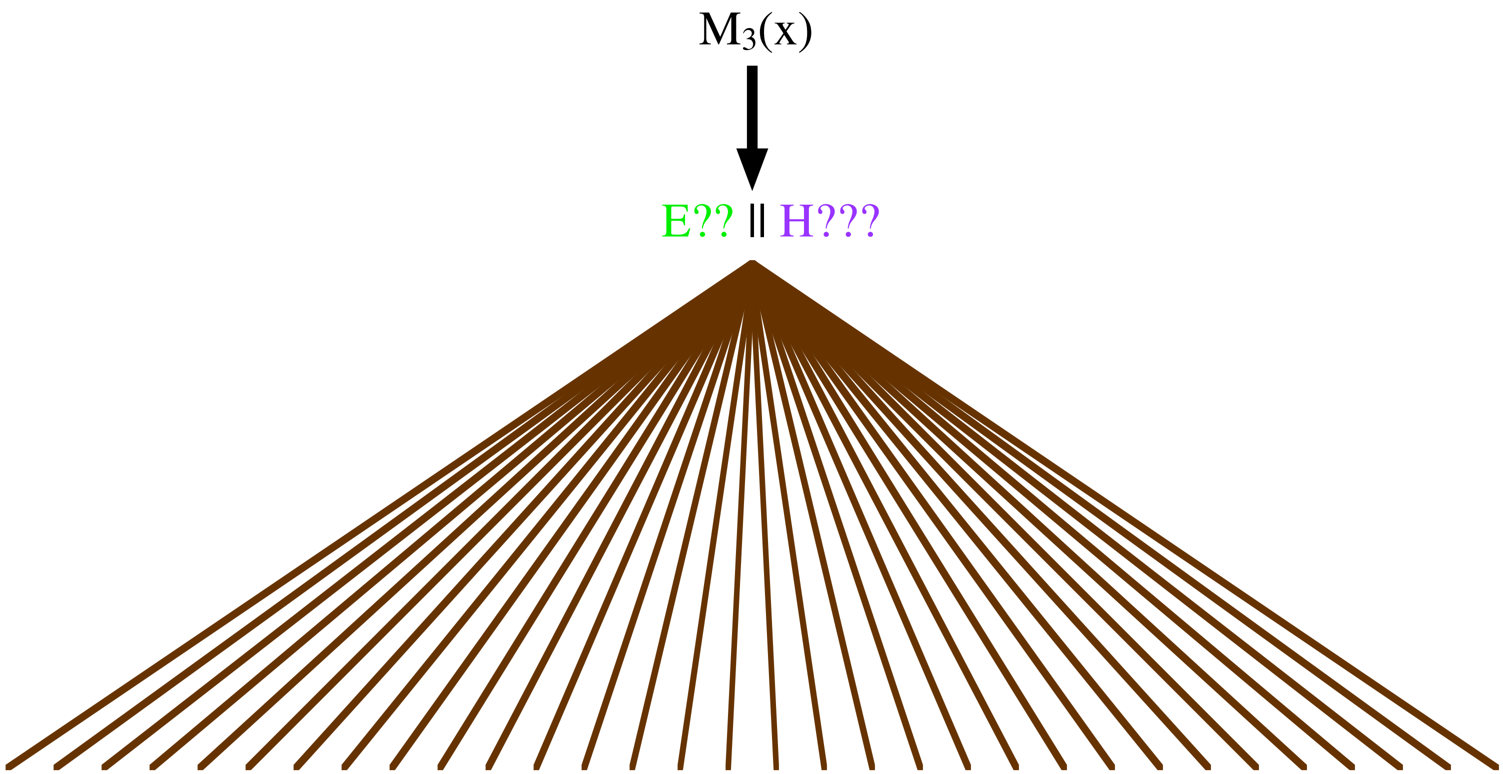
$E =$ easy oracle, Σ_j^{P} complete $\quad H =$ hard oracle, Σ_k^{P} complete $\quad j < k$



asking all questions simultaneously

$\mathrm{P}^{E_{s\text{-tt}} \| H_{r\text{-tt}}}$ = s queries to E and r queries to H in parallel

E = easy oracle, Σ_j^{P} complete $\qquad$ H = hard oracle, Σ_k^{P} complete $\qquad$ $j < k$



- Let $\mathrm{PF}^{A_{a\text{-tt}};B_{b\text{-tt}}}$ denote the class of functions recognized by polynomial-time Turing machines that ask a parallel queries to A followed by b parallel queries to B.

- Let $\mathrm{PF}^{A_{a\text{-tt}}\|B_{b\text{-tt}}}$ denote the class of functions recognized by polynomial-time Turing machines that ask a parallel queries to A simultaneous with b parallel queries to B.

- Let $\mathrm{P}^{A_{a\text{-tt}};B_{b\text{-tt}};C_{c\text{-tt}};D_{d\text{-tt}}}$ be the class of languages accepted by polynomial-time Turing machines that ask a queries to A, b queries to B, c queries to C and d queries to D in that order.

$\mathrm{PF}^{A_{a\text{-tt}}\|B_{b\text{-tt}}}$ is trivially contained in both $\mathrm{PF}^{A_{a\text{-tt}};B_{b\text{-tt}}}$ and $\mathrm{P}^{B_{b\text{-tt}};A_{a\text{-tt}}}$.

- Does not hurt to ask hard questions first

$$\mathrm{P}^{E_{s\text{-tt}};H_{r\text{-tt}}} \subseteq \mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}}$$

$$\mathrm{PF}^{E_{s\text{-tt}};H_{r\text{-tt}}} \subseteq \mathrm{PF}^{H_{r\text{-tt}};E_{s\text{-tt}}}$$

- For language classes, order does not matter

$$\mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}} \subseteq \mathrm{P}^{E_{s\text{-tt}};H_{r\text{-tt}}}$$

- For function classes, order matters unless PH collapses

$$\mathrm{PF}^{H_{r\text{-tt}};E_{s\text{-tt}}} \subseteq \mathrm{PF}^{E_{s\text{-tt}};H_{r\text{-tt}}} \Longrightarrow \mathrm{PH} \subseteq \Sigma^{\mathrm{P}}_{j+1}$$

where $j < k$, E is Σ^{P}_j complete and H is Σ^{P}_k-complete

prior & related works

- Hemaspaandra, Hempel & Wechsung 1995:

  Order of queries to 2 complete languages from the Boolean Hierarchy

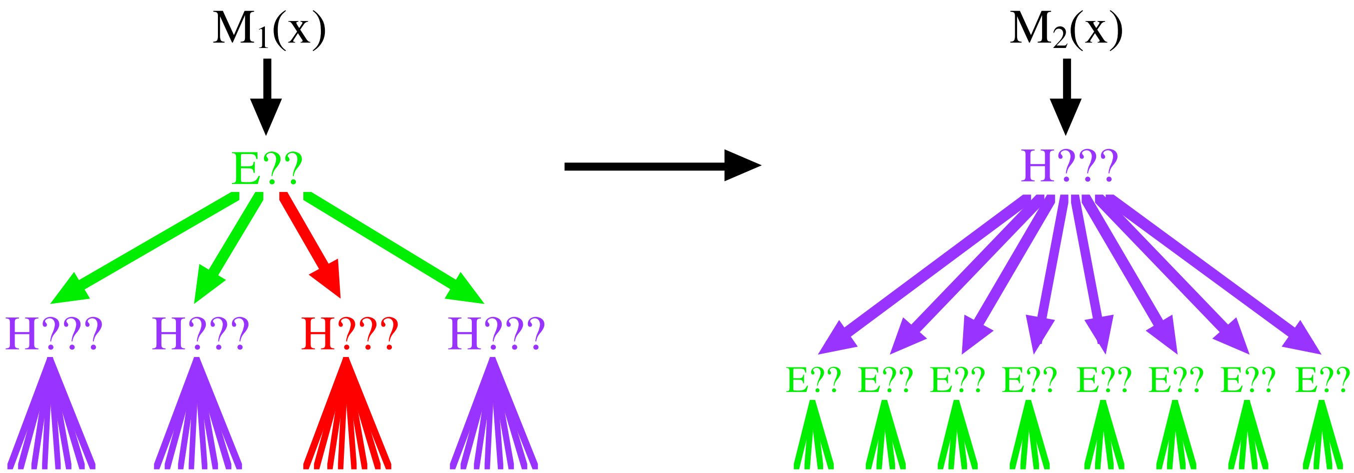
- Agrawal, Beigel & Thierauf 1996:

  Strengthened results on queries to complete languages from the Boolean Hierarchy. (Obtained independently from [HHW95].)

- Gasarch & McNicholl 1997(?):

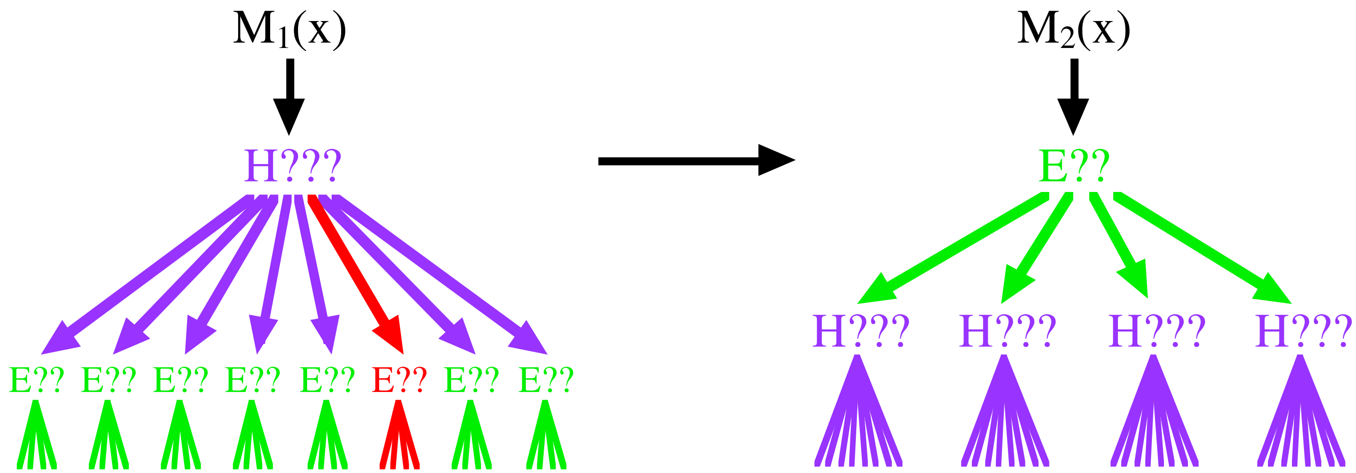
  Order of oracle queries in a recursion theoretic setting

Proof that $\mathrm{P}^{E_{s\text{-tt}};H_{r\text{-tt}}} \subseteq \mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}}$:

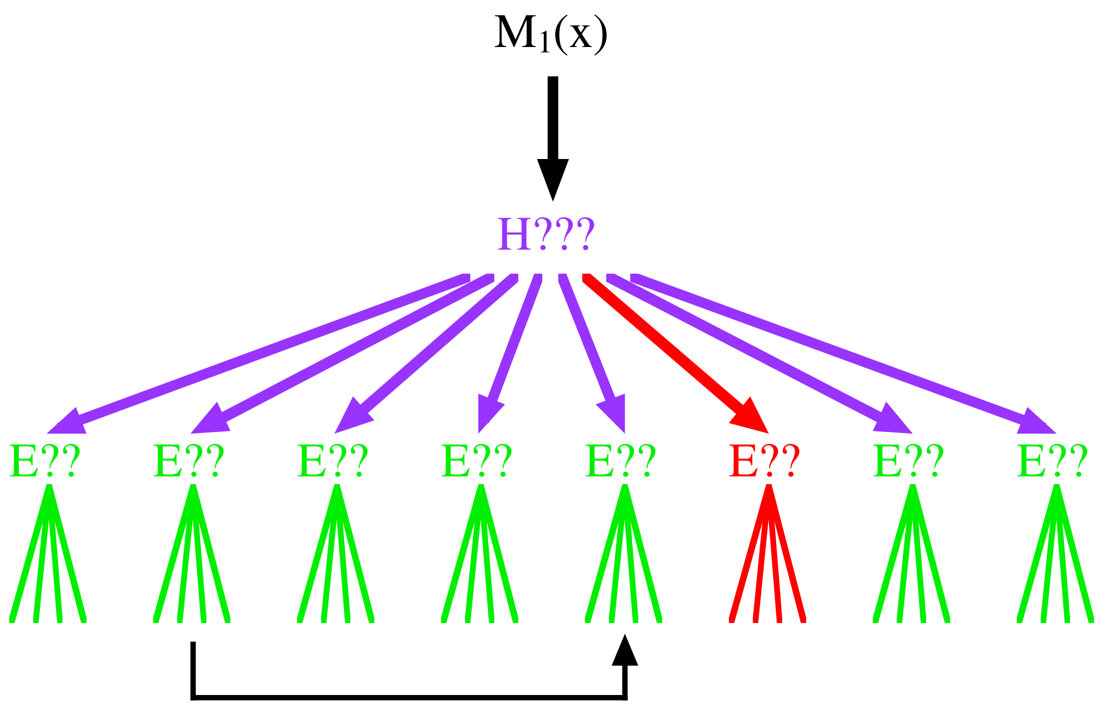


- M_2's ith query H: Is M_1's ith query to H answered YES?
- queries to E are the same
- in fact, $\mathrm{P}^{E_{s\text{-tt}};H_{r\text{-tt}}} \subseteq \mathrm{P}^{H_{r\text{-tt}} \| E_{s\text{-tt}}}$
- proof for function classes identical

Proof that $\mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}} \subseteq \mathrm{P}^{E_{s\text{-tt}};H_{r\text{-tt}}}$:



- Problem: Don't know which queries to E to ask
- Solution: Use the first set of queries to E
- Count the number of mind changes to the true path.



Path i to Path j forms a mind change if:

- Z_i = queries to H on Path i assumed to be answered YES.
- Z_j = queries to H on Path j assumed to be answered YES.
- $Z_i \subseteq Z_j \subseteq H$.
- $M_1(x)$ accepts on Path i and rejects on Path j or vice versa.

Finishing the mind change proof:

- paths beyond true path are not involved in mind changes
- maximum number of mind changes m ranges from 0 to $r - 1$
- m can be computed using r truth-table queries to H
- Whether $M_1(x)$ accepts on Path 0 can be computed using s queries to E
- m is odd: $M_1(x)$ accepts on true path iff $M_1(x)$ rejects on Path 0
- m is even: $M_1(x)$ accepts on true path iff $M_1(x)$ accepts on Path 0

We really proved that $\mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}} = \mathrm{P}^{E_{s\text{-tt}} \| H_{r\text{-tt}}}$.

How are the two classes $\mathrm{P}^{H_{a\text{-tt}};E_{b\text{-tt}}}$ and $\mathrm{P}^{H_{c\text{-tt}};E_{d\text{-tt}}}$ related?

- In the mind change proof, the s queries to E were used to determine whether $M_1(x)$ accepts or rejects on Path 0. This can be replaced by a single query to H.

- For all polynomial bounded s, $\mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}} \subseteq \mathrm{P}^{H_{(r+1)\text{-tt}}}$.

- Nested hierarchy:

$$\mathrm{P}^{H_{r\text{-tt}}} \subsetneq \mathrm{P}^{H_{r\text{-tt}} \| E_{1\text{-tt}}} \subsetneq \mathrm{P}^{H_{r\text{-tt}} \| E_{2\text{-tt}}} \subsetneq \cdots \subsetneq \mathrm{P}^{H_{r+1\text{-tt}}},$$

  unless the Polynomial Hierarchy collapses.

extensions: many rounds of queries

What happens if you have many rounds of truth-table queries to E and H?

- Easy queries can still be delayed:

$$\mathrm{P}^{H_{a\text{-tt}};E_{b\text{-tt}};H_{c\text{-tt}};E_{d\text{-tt}}} \subseteq \mathrm{P}^{H_{a\text{-tt}};H_{c\text{-tt}};E_{b\text{-tt}};E_{d\text{-tt}}}.$$

- Rounds of queries to the same oracle can be combined:

$$\mathrm{P}^{H_{a\text{-tt}};H_{c\text{-tt}};E_{b\text{-tt}};E_{d\text{-tt}}} \subseteq \mathrm{P}^{H_{r\text{-tt}};E_{s\text{-tt}}}$$

  where $r = (a+1)(c+1)$ and $s = (b+1)(d+1)$.

- Plus: $\mathrm{P}^{H_{r\text{-tt}}\|E_{s\text{-tt}}} \subseteq \mathrm{P}^{H_{a\text{-tt}};E_{b\text{-tt}};H_{c\text{-tt}};E_{d\text{-tt}}}$.

- Therefore, $\mathrm{P}^{H_{a\text{-tt}};E_{b\text{-tt}};H_{c\text{-tt}};E_{d\text{-tt}}} = \mathrm{P}^{H_{r\text{-tt}}\|E_{s\text{-tt}}}$.

Complexity of language classes characterized by the number of queries.

The order of the queries does not matter for language classes.

For function classes, the order of oracle queries is critical.

- We can still delay easy questions (same proof as language classes):

$$\mathrm{PF}^{E_s\text{-tt};H_r\text{-tt}} \subseteq \mathrm{PF}^{H_r\text{-tt};E_s\text{-tt}}$$

- We cannot delay hard questions unless PH collapses:

$$\mathrm{PF}^{H_r\text{-tt};E_s\text{-tt}} \subseteq \mathrm{PF}^{E_s\text{-tt};H_r\text{-tt}} \Longrightarrow \mathrm{PH} \subseteq \Sigma_{j+1}^{\mathrm{P}}$$

(Recall: $j < k$, E is Σ_j^{P} complete and H is Σ_k^{P} complete.)

- Proof uses the latest hard/easy argument [Buhrman & Fortnow, 1996] and tree pruning techniques [Beigel, Kummer & Stephan, 1995]

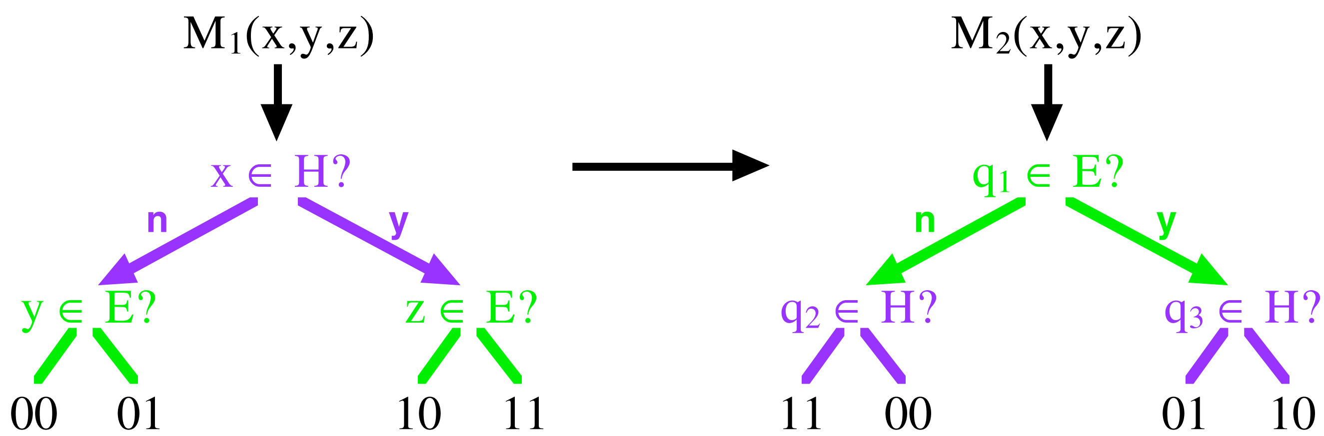
Let E be NP-complete, H be $\mathrm{NP}^{\mathrm{NP}}$-complete. Use 1 query to each oracle.

Candidate function in $\mathrm{PF}^{H_{\text{1-tt}};E_{\text{1-tt}}}$ but not in $\mathrm{PF}^{E_{\text{1-tt}};H_{\text{1-tt}}}$.

$$f(x,y,z) = \begin{cases} \left.\begin{array}{l} 00 \text{ if } x \notin H \text{ and } y \notin E \\ 01 \text{ if } x \notin H \text{ and } y \in E \end{array}\right\} = H(x)E(y) \text{ if } x \notin H \\ \\ \left.\begin{array}{l} 10 \text{ if } x \in H \text{ and } z \notin E \\ 11 \text{ if } x \in H \text{ and } z \in E \end{array}\right\} = H(x)E(z) \text{ if } x \in H \end{cases}$$

- $H(x)$, $E(y)$ and $E(z)$ are characteristic functions
- $H(x)E(y)$ means concatenation
- $f(x,y,z)$ is easily computable in $\mathrm{PF}^{H_{\text{1-tt}};E_{\text{1-tt}}}$
- Prove $f(x,y,z) \in \mathrm{PF}^{E_{\text{1-tt}};H_{\text{1-tt}}} \Longrightarrow \overline{H} \subseteq \mathrm{NP}^{\mathrm{NP}} \Longrightarrow \mathrm{PH} \subseteq \mathrm{NP}^{\mathrm{NP}}$.

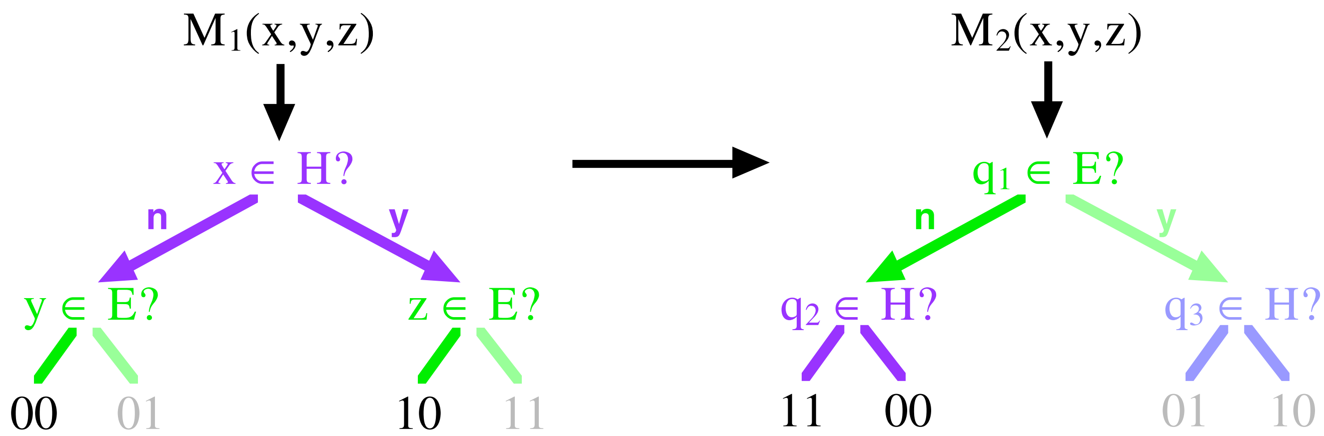
A $\mathrm{PF}^{H_{1\text{-tt}};E_{1\text{-tt}}}$ machine and a $\mathrm{PF}^{E_{1\text{-tt}};H_{1\text{-tt}}}$ machine which compute $f(x, y, z)$



- Both machines compute $f(x, y, z)$ correctly
- Construct an $\mathrm{NP}^{\mathrm{NP}}$ machine for $\overline{H}$ which is $\mathrm{coNP}^{\mathrm{NP}}$ complete
- Use NP oracle to answer all queries to E
- Let OUT_1 and OUT_2 be the possible outputs of M_1 and M_2 after queries to E are answered (some paths are eliminated)
- Example: $y \notin E$, $z \notin E$ and $q_1 \notin E$

$$\mathrm{OUT}_1 = \{00, 10\} \qquad \mathrm{OUT}_2 = \{00, 11\}$$

x is easy if there exists y and z, $|y| = |z| \le |x|^k$, such that $\mathrm{OUT}_1 \ne \mathrm{OUT}_2$

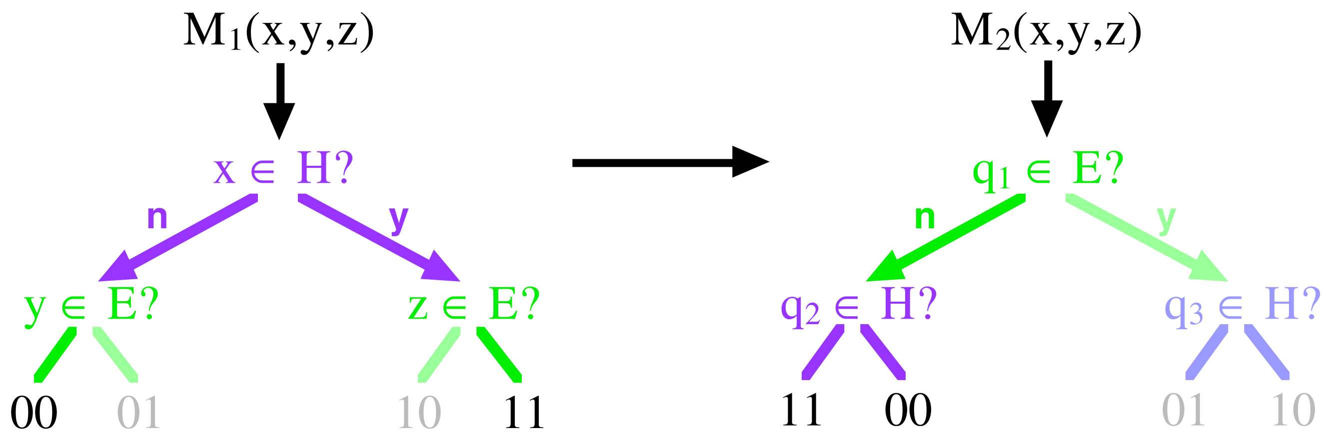


If $y \notin E$, $z \notin E$ and $q_1 \notin E$, then $\mathrm{OUT}_1 \cap \mathrm{OUT}_2 = \{00\} \Longrightarrow H(x) = 0$

If x is easy, this $\mathrm{NP}^{\mathrm{NP}}$ algorithm recognizes $\overline{H}$

- Guess y and z with length $\le |x|^k$
- Compute OUT_1 and OUT_2 by simulating $M_1(x, y, z)$ and $M_2(x, y, z)$ using the NP oracle to answer all queries to E
- If $\mathrm{OUT}_1 = \mathrm{OUT}_2$, reject
- Otherwise, $f(x, y, z) = \mathrm{OUT}_1 \cap \mathrm{OUT}_2$ and the first bit of $f(x, y, z)$ is $H(x)$

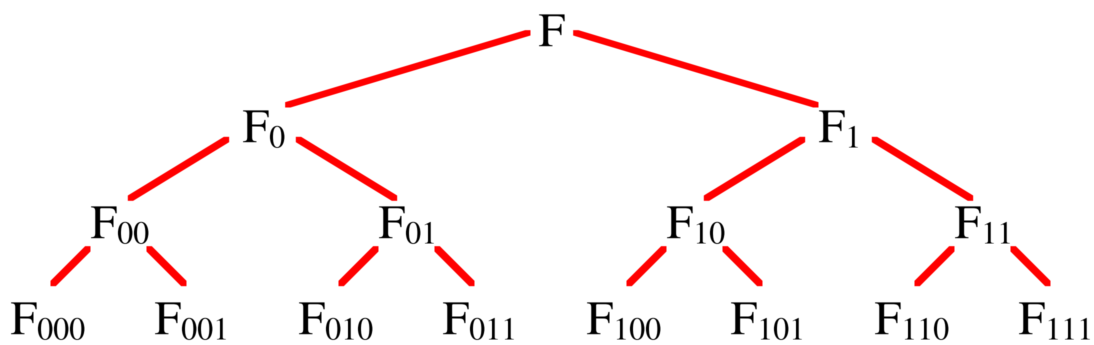
x is hard if for all y and z, $|y| = |z| \leq |x|^k$, $\mathrm{OUT}_1 = \mathrm{OUT}_2$



If $y \notin E$, $z \in E$ and $q_1 \notin E$, $\mathrm{OUT}_1 = \mathrm{OUT}_2 = \{00, 11\} \Longrightarrow E(y)E(z) = 01$

- with 1 query to E, the outcome of two queries were determined.
- with 0 queries to E, we still have $E(y)E(z) \in \{01, 10\}$
- 2 out of 4 possibilities for $E(y)E(z)$ eliminated using 0 queries!
- this is enough to prove that SAT $\in$ P.

self-reduction tree for SAT



- $F_{w0} = F_w$ with first variable replaced by FALSE
- $F_{w1} = F_w$ with first variable replaced by TRUE
- $F_w \in \mathrm{SAT} \iff (F_{w0} \in \mathrm{SAT}) \vee (F_{w1} \in \mathrm{SAT})$
- use Beigel-Kummer-Stephan (BKS) tree pruning procedure:

  Given 4 formulas, find 1 to drop "safely"

BKS tree pruning

Given $Q = \{F_1, F_2, F_3, F_4\}$, find $i \in \{1, 2, 3, 4\}$ such that

$$Q \cap \mathrm{SAT} \neq \emptyset \iff (Q - F_i) \cap \mathrm{SAT} \neq \emptyset$$

(I.e., F_i is not the only element of Q in SAT.)

Since E is NP-complete, we can construct y and z such that

$$y \in E \iff (F_3 \in \mathrm{SAT}) \vee (F_4 \in \mathrm{SAT})$$

$$z \in E \iff (F_2 \in \mathrm{SAT}) \vee (F_4 \in \mathrm{SAT})$$

- $Q \cap \mathrm{SAT} = \{F_1\} \Longrightarrow E(y)E(z) = 00.$ if $E(y)E(z) \neq 00$, drop F_1.
- $Q \cap \mathrm{SAT} = \{F_2\} \Longrightarrow E(y)E(z) = 01.$ if $E(y)E(z) \neq 01$, drop F_2.
- $Q \cap \mathrm{SAT} = \{F_3\} \Longrightarrow E(y)E(z) = 10.$ if $E(y)E(z) \neq 10$, drop F_3.
- $Q \cap \mathrm{SAT} = \{F_4\} \Longrightarrow E(y)E(z) = 11.$ if $E(y)E(z) \neq 11$, drop F_4.

hard case: revisited

Need to show that $\overline{H} \in \mathrm{NP}^{\mathrm{NP}}$. Rewrite $\overline{H}$ as:

$$\overline{H} = \{x \mid (\forall u, |u| = |x|)[\; g(x,u) \in \mathrm{SAT} \;]\}$$

P^{NP} algorithm for $\overline{H}$ assuming x is hard:

- construct an NP machine N.
  1. input x
  2. Guess u, compute $F = g(x,u)$
  3. use BKS tree pruning to find witness for $F \in \mathrm{SAT}$.
  4. if witness is found, reject
  5. no witness after pruning, accept
- ask NP oracle whether $x \in L(N)$
- $x \notin L(N)$, accept /* since satisfying assignment found for each $g(x,u)$ */
- $x \in L(N)$, reject

Problem: we don't know if x is easy or hard.

Solution: Run easy and hard algorithms in parallel

Sanity check on combined algorithm

- $x \notin \overline{H}$ and x is easy
  - easy algorithm: works correctly and rejects.
  - hard algorithm: $N(x)$ cannot find a satisfying assignment for $g(x,u)$ for some u. Machine N accepts, hard algorithm rejects.
- $x \notin \overline{H}$ and x is hard
  - easy algorithm: cannot find y and z such that $\mathrm{OUT}_1 \neq \mathrm{OUT}_2$. Rejects.
  - hard algorithm: works correctly and rejects.

- $x \in \overline{H}$ and x is easy
  - easy algorithm: works correctly and accepts.
  - hard algorithm: $N(x)$ might be "lucky" and find satisfying assignments for $g(x, u)$, for every u. Algorithm might accept.
- $x \in \overline{H}$ and x is hard
  - easy algorithm: cannot find y and z such that $\mathrm{OUT}_1 \neq \mathrm{OUT}_2$. Rejects.
  - hard algorithm: works correctly and accepts.

So, we have an $\mathrm{NP}^{\mathrm{NP}}$ algorithm for $\overline{H}$.
Since H is $\mathrm{NP}^{\mathrm{NP}}$ complete, $\mathrm{NP}^{\mathrm{NP}} = \mathrm{coNP}^{\mathrm{NP}}$ and PH collapses to $\mathrm{NP}^{\mathrm{NP}}$.

Let H be Σ_k^{P} complete and E be Σ_j^{P} complete, where $j < k$.

Does the order of oracle access matter?

- Language Classes: NO
- Function Classes: YES