P versus NP: Approaches, Rebuttals, and Does It Matter?

Neil Immerman

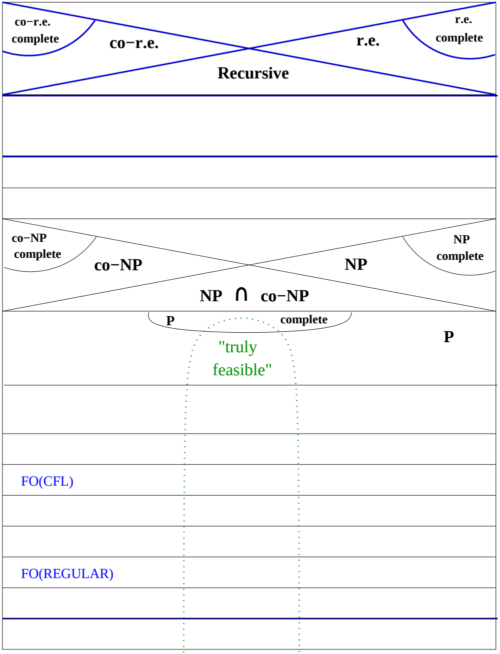
www.cs.umass.edu/∼immerman

Spike of attention to P vs. NP problem, Aug. 2010

"Deolalikar claimed that he had tamed the wildness of algorithms and shown that P indeed doesnt equal NP. Within a few hours of his e-mail, the paper got an impressive endorsement: 'This appears to be a relatively serious claim to have solved P versus NP,' emailed Stephen Cook of the University of Toronto, the scientist who had initially formulated the question. That evening, a blogger posted Deolalikar's paper. And the next day, long before researchers had had time to examine the 103-page paper in detail, the recommendation site Slashdot picked it up, sending a fire hose of tens of thousands of readers and dozens of journalists to the paper."

Julie Rehmeyer, *Science News*, Sept. 9, 2010

$$P = \bigcup_{k=1}^{\infty} \mathrm{DTIME}[n^k]$$

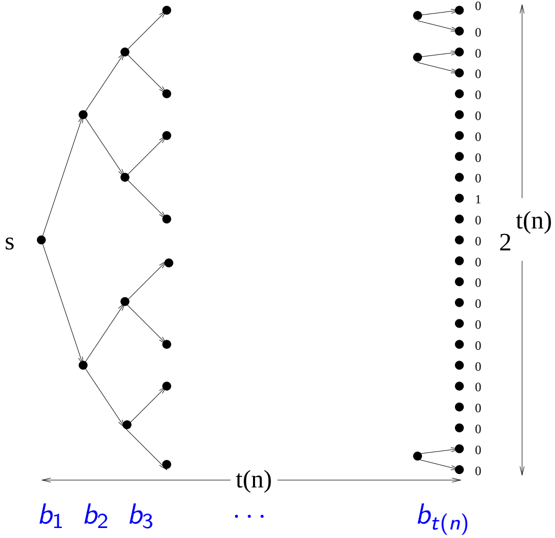


P is a good mathematical wrapper for "truly feasible".

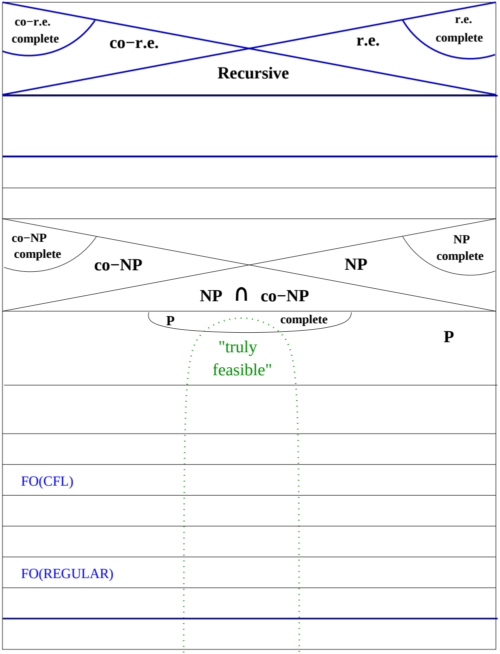
NTIME[$t(n)$]: a mathematical fiction

input w

$|w| = n$

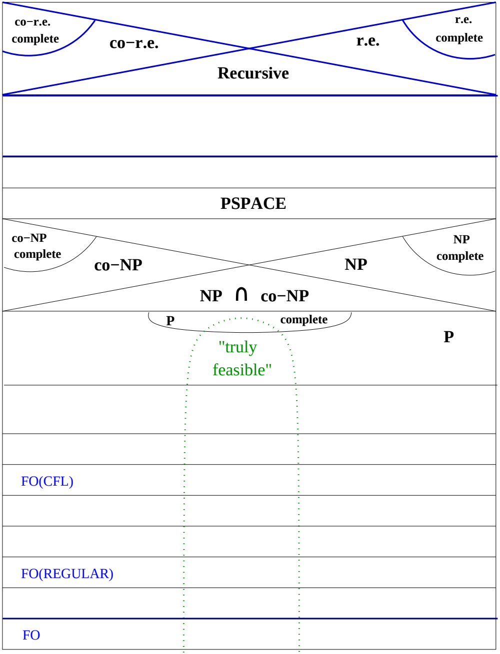


$$\mathrm{NP} \quad = \quad \bigcup_{k=1}^{\infty} \mathrm{NTIME}[n^k]$$



Many optimization problems we want to solve are NP complete.

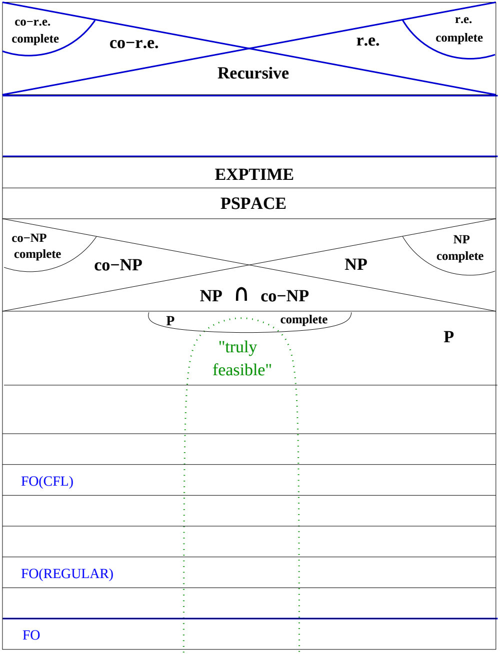
$$NP = \bigcup_{k=1}^{\infty} NTIME[n^k]$$



Many optimization problems we want to solve are NP complete.

$$\mathrm{NP} = \bigcup_{k=1}^{\infty} \mathrm{NTIME}[n^k]$$

Many optimization problems we want to solve are NP complete.



Query $\mapsto$ $\boxed{\textbf{Computation}}$ $\mapsto$ **Answer**

$q_1\ q_2\ \cdots\ q_n$ $\qquad\qquad$ $a_1\ a_2\ \cdots\ a_i\ \cdots\ a_m$

Descriptive Complexity

Query $\mapsto$ $\boxed{\textbf{Computation}}$ $\mapsto$ **Answer**

$q_1\ q_2\ \cdots\ q_n$ $\qquad\qquad$ $a_1\ a_2\ \cdots\ a_i\ \cdots\ a_m$

Restrict attention to the complexity of computing individual bits of the output, i.e., **decision problems**.

$$\begin{array}{ccccc} \textbf{Query} & & & & \textbf{Answer} \\ q_1\ q_2\ \cdots\ q_n & \mapsto & \boxed{\textbf{Computation}} & \mapsto & a_1\ a_2\ \cdots\ a_i\ \cdots\ a_m \\ & & & & \cdots\ S_i\ \cdots \end{array}$$

Restrict attention to the complexity of computing individual bits of the output, i.e., **decision problems**.

How hard is it to **check** if input has property S ?

Query $\mapsto$ $\boxed{\textbf{Computation}}$ $\mapsto$ **Answer**

$q_1 \; q_2 \; \cdots \; q_n$ $\qquad$ $a_1 \; a_2 \; \cdots \; a_i \; \cdots \; a_m$

$\cdots \; S_i \; \cdots$

Restrict attention to the complexity of computing individual bits of the output, i.e., **decision problems**.

How hard is it to **check** if input has property S ?

How rich a language do we need to **express** property S?

Descriptive Complexity

$$\begin{array}{ccccc} \textbf{Query} & & & & \textbf{Answer} \\ q_1\ q_2\ \cdots\ q_n & \mapsto & \boxed{\textbf{Computation}} & \mapsto & a_1\ a_2\ \cdots\ a_i\ \cdots\ a_m \\ & & & & \cdots\ S_i\ \cdots \end{array}$$

Restrict attention to the complexity of computing individual bits of the output, i.e., **decision problems**.

How hard is it to **check** if input has property S ?

How rich a language do we need to **express** property S?

There is a constructive isomorphism between these two approaches.

Interpret Input as Finite Logical Structure

Graph $\quad G = (\{v_1, \ldots, v_n\}, E, s, t)$

Binary String

$$\mathcal{A}_w = (\{p_1, \ldots, p_8\}, S)$$
$$S = \{p_2, p_5, p_7, p_8\}$$
$$w = 01001011$$

Vocabularies: $\tau_g = (E^2, s, t)$, $\quad \tau_s = (S^1)$

First-Order Logic

input symbols:	from τ
variables:	$x, y, z, \dots$
boolean connectives:	$\land, \lor, \neg$
quantifiers:	$\forall, \exists$
numeric symbols:	$=, \leq, +, \times, \min, max$

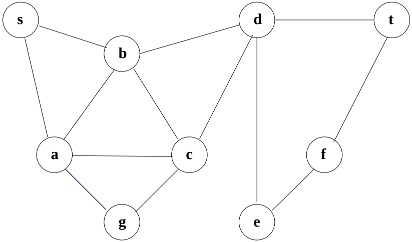
$$\alpha \quad \equiv \quad \forall x \exists y (E(x,y)) \quad \in \quad \mathcal{L}(\tau_g)$$

$$\beta \quad \equiv \quad \exists x \forall y (x \leq y \land S(x)) \quad \in \quad \mathcal{L}(\tau_s)$$

$$\beta \quad \equiv \quad S(\min) \quad \in \quad \mathcal{L}(\tau_s)$$

Second-Order Logic

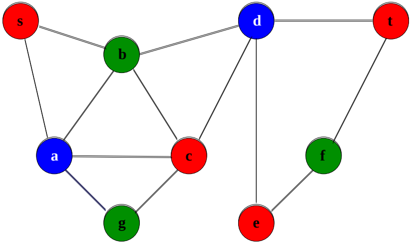
$$\Phi_{3-\text{color}} \equiv \exists R^1 G^1 B^1 \forall x\, y\, ((R(x) \vee G(x) \vee B(x)) \wedge (E(x,y) \to (\neg(R(x) \wedge R(y)) \wedge \neg(G(x) \wedge G(y)) \wedge \neg(B(x) \wedge B(y)))))$$

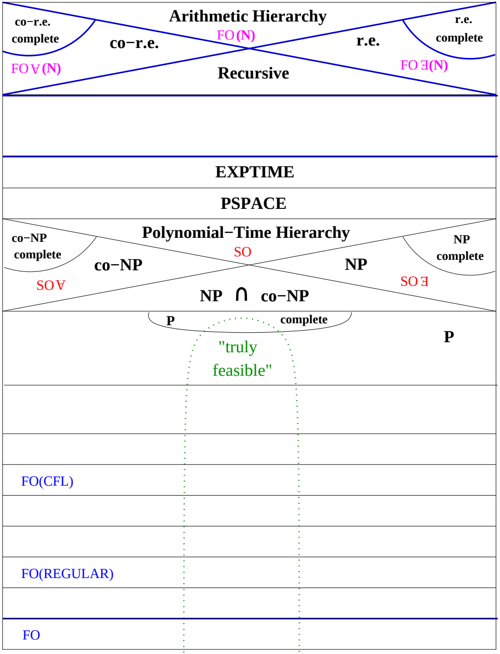


Second-Order Logic

Fagin's Theorem: NP $=$ SO$\exists$

$$\Phi_{3-\text{color}} \equiv \exists\, R^1\, G^1\, B^1\, \forall x\, y\, ((R(x) \vee G(x) \vee B(x)) \;\wedge$$
$$(E(x,y) \rightarrow (\neg(R(x) \wedge R(y)) \wedge \neg(G(x) \wedge G(y))$$
$$\wedge \quad \neg(B(x) \wedge B(y)))))$$





Arithmetic Hierarchy
co−r.e. complete
co−r.e.
FO (N)
r.e.
r.e. complete
FO ∀(N)
Recursive
FO ∃(N)
EXPTIME
PSPACE
Polynomial−Time Hierarchy
co−NP complete
SO
co−NP
NP
NP complete
SO ∀
SO ∃
NP ∩ co−NP
P complete
P
"truly feasible"
FO(CFL)
FO(REGULAR)
FO

Addition is First-Order

$$Q_+ : \mathrm{STRUC}[\tau_{AB}] \to \mathrm{STRUC}[\tau_s]$$

$$\begin{array}{lccccccc}
A & & a_1 & a_2 & \ldots & a_{n-1} & a_n \\
B & + & b_1 & b_2 & \ldots & b_{n-1} & b_n \\
\cline{3-7}
S & & s_1 & s_2 & \ldots & s_{n-1} & s_n
\end{array}$$

Addition is First-Order

$$Q_+ : \mathrm{STRUC}[\tau_{AB}] \to \mathrm{STRUC}[\tau_s]$$

$$\begin{array}{lcccccc}
A & & a_1 & a_2 & \ldots & a_{n-1} & a_n \\
B & + & b_1 & b_2 & \ldots & b_{n-1} & b_n \\
\hline
S & & s_1 & s_2 & \ldots & s_{n-1} & s_n
\end{array}$$

$$\begin{aligned}
C(i) \quad \equiv \quad & (\exists j > i)\Big(A(j) \wedge B(j) \;\wedge \\
& \qquad (\forall k.j > k > i)(A(k) \vee B(k))\Big)
\end{aligned}$$

Addition is First-Order

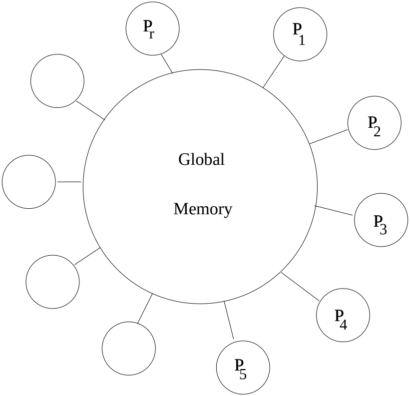
$$Q_+ : \mathrm{STRUC}[\tau_{AB}] \to \mathrm{STRUC}[\tau_s]$$

$$\begin{array}{lcccccc}
A & & a_1 & a_2 & \dots & a_{n-1} & a_n \\
B & + & b_1 & b_2 & \dots & b_{n-1} & b_n \\
\hline
S & & s_1 & s_2 & \dots & s_{n-1} & s_n
\end{array}$$

$$\begin{aligned}
C(i) \quad &\equiv \quad (\exists j > i)\Big(A(j) \wedge B(j) \ \wedge \\
&\qquad\qquad (\forall k.j > k > i)(A(k) \vee B(k))\Big) \\
Q_+(i) \quad &\equiv \quad A(i) \ \oplus \ B(i) \ \oplus \ C(i)
\end{aligned}$$

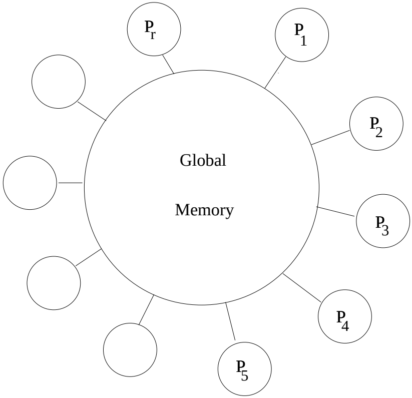
Parallel Machines:

$\text{CRAM}[t(n)] = \text{CRCW-PRAM-TIME}[t(n)]\text{-HARD}[n^{O(1)}]$



$\text{CRAM}[t(n)] \; = \text{CRCW-PRAM-TIME}[t(n)]\text{-HARD}[n^{O(1)}]$

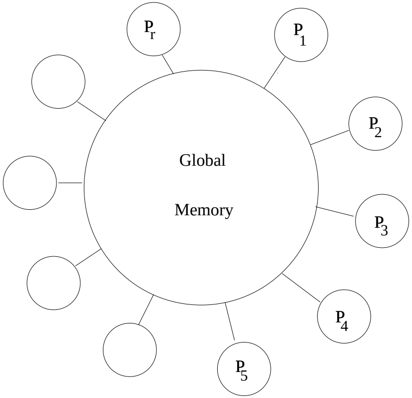
Assume array $A[x] : x = 1, \ldots, r$ in memory.



$\mathrm{CRAM}[t(n)] \;=\mathrm{CRCW\text{-}PRAM\text{-}TIME}[t(n)]\text{-}\mathrm{HARD}[n^{O(1)}]$

Assume array $A[x] : x = 1, \ldots, r$ in memory.

$$\forall x(A(x)) \;\equiv\; \textbf{write}(1);\; \text{proc } p_i : \textbf{if } (A[i] = 0) \textbf{ then } \; \textbf{write}(0)$$



FO

=

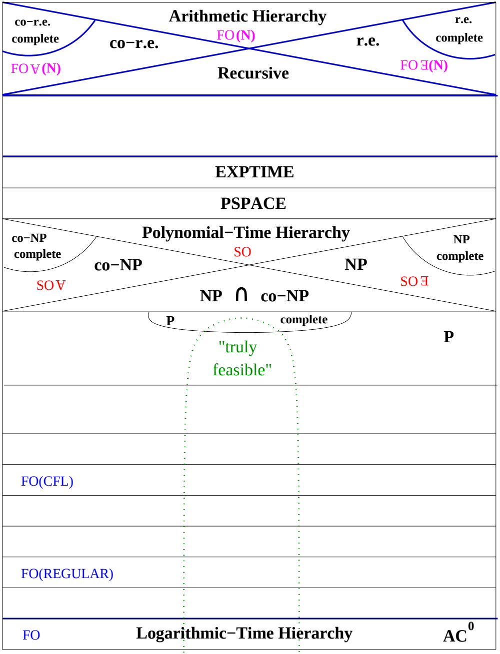
CRAM[1]

=

AC^0

=

Logarithmic-Time Hierarchy



Inductive Definitions

$$E^\star(x,y) \quad \equiv \quad x = y \ \vee \ E(x,y) \ \vee \ \exists z(E^\star(x,z) \wedge E^\star(z,y))$$

Inductive Definitions

$$E^\star(x,y) \equiv x = y \lor E(x,y) \lor \exists z(E^\star(x,z) \land E^\star(z,y))$$

$$\varphi_{tc}(R,x,y) \equiv x = y \lor E(x,y) \lor \exists z(R(x,z) \land R(z,y))$$

Inductive Definitions

$$E^\star(x,y) \quad\equiv\quad x = y \ \vee\ E(x,y) \ \vee\ \exists z(E^\star(x,z) \wedge E^\star(z,y))$$

$$\varphi_{tc}(R,x,y) \quad\equiv\quad x = y \ \vee\ E(x,y) \ \vee\ \exists z(R(x,z) \wedge R(z,y))$$

$$G \in \mathrm{REACH} \quad\Leftrightarrow\quad G \models (\mathrm{LFP}\varphi_{tc})(s,t)$$

Inductive Definitions

$$
\begin{aligned}
E^\star(x,y) &\equiv x = y \ \vee\ E(x,y) \ \vee\ \exists z(E^\star(x,z) \wedge E^\star(z,y)) \\
\varphi_{tc}(R,x,y) &\equiv x = y \ \vee\ E(x,y) \ \vee\ \exists z(R(x,z) \wedge R(z,y)) \\
G \in \mathrm{REACH} &\Leftrightarrow G \models (\mathrm{LFP}\varphi_{tc})(s,t)
\end{aligned}
$$

Thus, $\mathrm{REACH} \in \mathrm{IND}[\log n]$.

Inductive Definitions

$$E^\star(x,y) \equiv x = y \lor E(x,y) \lor \exists z(E^\star(x,z) \land E^\star(z,y))$$

$$\varphi_{tc}(R,x,y) \equiv x = y \lor E(x,y) \lor \exists z(R(x,z) \land R(z,y))$$

$$G \in \mathrm{REACH} \Leftrightarrow G \models (\mathrm{LFP}\varphi_{tc})(s,t)$$

Thus, $\mathrm{REACH} \in \mathrm{IND}[\log n]$.

Next, we'll show that $\mathrm{REACH} \in \mathrm{FO}[\log n]$.

$\varphi_{tc}(R, x, y) \equiv x = y \vee E(x, y) \vee \exists z\,(R(x, z) \wedge R(z, y))$

1. Dummy universal quantification for base case:

$$\begin{aligned}\varphi_{tc}(R, x, y) &\equiv (\forall z.M_1)(\exists z)(R(x, z) \wedge R(z, y)) \\ M_1 &\equiv \neg(x = y \vee E(x, y))\end{aligned}$$

$\varphi_{tc}(R,x,y) \equiv x = y \lor E(x,y) \lor \exists z\,(R(x,z) \land R(z,y))$

1. Dummy universal quantification for base case:

$$\begin{aligned}\varphi_{tc}(R,x,y) &\equiv (\forall z.M_1)(\exists z)(R(x,z) \land R(z,y))\\ M_1 &\equiv \neg(x = y \lor E(x,y))\end{aligned}$$

2. Using $\forall$, replace two occurrences of R with one:

$$\begin{aligned}\varphi_{tc}(R,x,y) &\equiv (\forall z.M_1)(\exists z)(\forall uv.M_2)R(u,v)\\ M_2 &\equiv (u = x \land v = z) \lor (u = z \land v = y)\end{aligned}$$

$\varphi_{tc}(R,x,y) \equiv x = y \vee E(x,y) \vee \exists z\,(R(x,z) \wedge R(z,y))$

1. Dummy universal quantification for base case:

$$\begin{aligned} \varphi_{tc}(R,x,y) &\equiv (\forall z.M_1)(\exists z)(R(x,z) \wedge R(z,y)) \\ M_1 &\equiv \neg(x = y \vee E(x,y)) \end{aligned}$$

2. Using $\forall$, replace two occurrences of R with one:

$$\begin{aligned} \varphi_{tc}(R,x,y) &\equiv (\forall z.M_1)(\exists z)(\forall uv.M_2)R(u,v) \\ M_2 &\equiv (u = x \wedge v = z) \vee (u = z \wedge v = y) \end{aligned}$$

3. Requantify x and y.

$$M_3 \equiv (x = u \wedge y = v)$$

$$\varphi_{tc}(R,x,y) \equiv [\,(\forall z.M_1)(\exists z)(\forall uv.M_2)(\exists xy.M_3)\,]\,R(x,y)$$

Every FO inductive definition is equivalent to a quantifier block.

$\mathrm{QB}_{tc} \quad \equiv \quad [(\forall z.M_1)(\exists z)(\forall uv.M_2)(\forall xy.M_3)]$

$$\varphi_{tc}(R,x,y) \quad \equiv \quad [(\forall z.M_1)(\exists z)(\forall uv.M_2)(\exists xy.M_3)]R(x,y)$$

$\mathrm{QB}_{tc} \quad \equiv \quad [(\forall z.M_1)(\exists z)(\forall uv.M_2)(\forall xy.M_3)]$

$$\varphi_{tc}(R,x,y) \quad \equiv \quad [(\forall z.M_1)(\exists z)(\forall uv.M_2)(\exists xy.M_3)]R(x,y)$$

$$\varphi_{tc}(R,x,y) \quad \equiv \quad [\mathrm{QB}_{tc}]R(x,y)$$

$\mathrm{QB}_{tc} \equiv [(\forall z.M_1)(\exists z)(\forall uv.M_2)(\forall xy.M_3)]$

$$\varphi_{tc}(R,x,y) \equiv [(\forall z.M_1)(\exists z)(\forall uv.M_2)(\exists xy.M_3)]R(x,y)$$

$$\varphi_{tc}(R,x,y) \equiv [\mathrm{QB}_{tc}]R(x,y)$$

$$\varphi^r_{tc}(\emptyset) \equiv [\mathrm{QB}_{tc}]^r(\mathbf{false})$$

$\mathrm{QB}_{tc} \quad \equiv \quad [(\forall z.M_1)(\exists z)(\forall uv.M_2)(\forall xy.M_3)]$

$$\varphi_{tc}(R,x,y) \equiv [(\forall z.M_1)(\exists z)(\forall uv.M_2)(\exists xy.M_3)]R(x,y)$$

$$\varphi_{tc}(R,x,y) \equiv [\mathrm{QB}_{tc}]R(x,y)$$

$$\varphi_{tc}^{r}(\emptyset) \equiv [\mathrm{QB}_{tc}]^{r}(\mathbf{false})$$

Thus, for any structure $\mathcal{A} \in \mathrm{STRUC}[\tau_g]$,

$$\begin{aligned}\mathcal{A} \in \mathrm{REACH} &\Leftrightarrow \mathcal{A} \models (\mathrm{LFP}\varphi_{tc})(s,t)\\ &\Leftrightarrow \mathcal{A} \models ([\mathrm{QB}_{tc}]^{\lceil 1+\log \|\mathcal{A}\| \rceil}\,\mathbf{false})(s,t)\end{aligned}$$

$$
\begin{aligned}
\mathrm{CRAM}[t(n)] &= \text{concurrent parallel random access machine;} \\
&\quad\ \text{polynomial hardware, parallel time } O(t(n)) \\[2ex]
\mathrm{IND}[t(n)] &= \text{first-order, depth } t(n) \text{ inductive definitions} \\[2ex]
\mathrm{FO}[t(n)] &= t(n) \text{ repetitions of a block of restricted quantifiers:} \\[2ex]
\mathrm{QB} &= [(Q_1 x_1 . M_1) \cdots (Q_k x_k . M_k)]; \quad M_i \text{ quantifier-free} \\[2ex]
\varphi_n &= \underbrace{[\mathrm{QB}][\mathrm{QB}] \cdots [\mathrm{QB}]}_{t(n)} M_0
\end{aligned}
$$

parallel time = inductive depth = QB iteration

Thm: For all constructible, polynomially bounded $t(n)$,

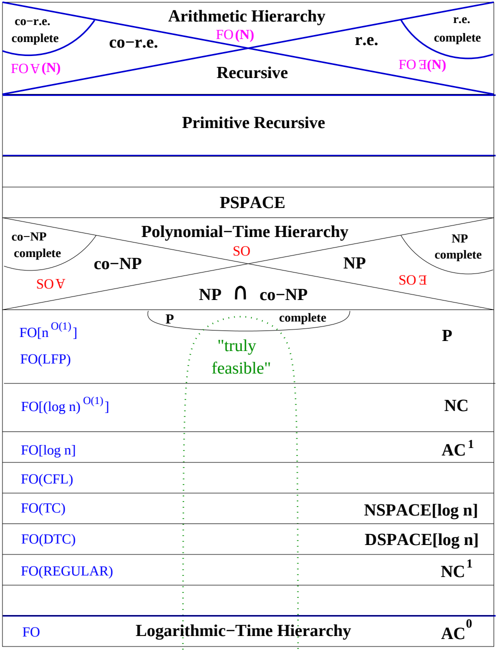
$$\mathrm{CRAM}[t(n)] \;=\; \mathrm{IND}[t(n)] \;=\; \mathrm{FO}[t(n)]$$

Thm: For all $t(n)$, even beyond polynomial,

$$\mathrm{CRAM}[t(n)] \quad = \quad \mathrm{FO}[t(n)]$$

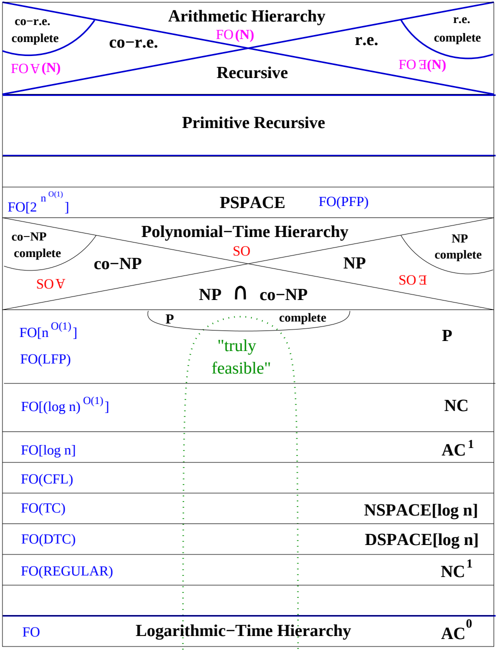
For $t(n)$ poly bdd,

$$\mathrm{CRAM}[t(n)] = \mathrm{IND}[t(n)] = \mathrm{FO}[t(n)]$$



For all $t(n)$,

$$\mathrm{CRAM}[t(n)] = \mathrm{FO}[t(n)]$$



Thm: For $v = 1, 2, \ldots, \quad \mathrm{DSPACE}[n^v] \;=\; \mathrm{VAR}[v+1]$

Number of variables corresponds to amount of hardware.

Since variables range over a universe of size n, a constant number of variables can specify a polynomial number of gates:

A bounded number of variables corresponds to polynomially much hardware.

Key Issue: Parallel Time versus Amount of Hardware

Key Issue: Parallel Time versus Amount of Hardware

- We would love to understand this tradeoff.

Key Issue: Parallel Time versus Amount of Hardware

- We would love to understand this tradeoff.
- Is there such a thing as an inherently sequential problem? No one knows.

Key Issue: Parallel Time versus Amount of Hardware

- We would love to understand this tradeoff.
- Is there such a thing as an inherently sequential problem? No one knows.
- Same tradeoff as number of variables vs. number of iterations of a quantifier block.

Key Issue: Parallel Time versus Amount of Hardware

- We would love to understand this tradeoff.
- Is there such a thing as an inherently sequential problem? No one knows.
- Same tradeoff as number of variables vs. number of iterations of a quantifier block.

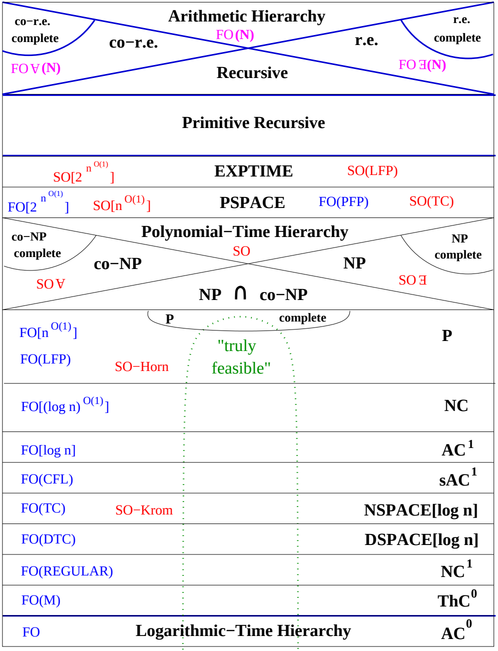
- One second-order variable can name 2^n gates.

Key Issue: Parallel Time versus Amount of Hardware

- We would love to understand this tradeoff.
- Is there such a thing as an inherently sequential problem? No one knows.
- Same tradeoff as number of variables vs. number of iterations of a quantifier block.

- One second-order variable can name 2^n gates.
- Thus, $\mathrm{SO}[t(n)] \; = \; \mathrm{CRAM\text{-}HARD}[t(n), 2^{n^{O(1)}}]$.

$$\mathrm{SO}[t(n)] = \text{CRAM-HARD}[t(n), 2^{n^{O(1)}}]$$



Recent Breakthroughs in Descriptive Complexity

Theorem [Ben Rossman] Any first-order formula with any numeric relations $(\leq, +, \times, \ldots)$ that means "I have a clique of size k" must have at least $k/4$ variables.

Creative new proof idea using Håstad's Switching Lemma gives the essentially optimal bound.

This lower bound is for a fixed formula, if it were for a sequence of polynomially-sized formulas, i.e., a fixed-point formula, it would follow that $\mathrm{CLIQUE} \notin \mathrm{P}$ and thus $\mathrm{P} \neq \mathrm{NP}$.

Best previous bounds:

- k variables necessary and sufficient without ordering or other numeric relations [I 1980].
- Nothing was known with ordering except for the trivial fact that 2 variables are not enough.

Recent Breakthroughs in Descriptive Complexity

Theorem [Martin Grohe] Fixed-Point Logic with Counting captures Polynomial Time on all classes of graphs with excluded minors.

Grohe proves that for every class of graphs with excluded minors, there is a constant k such that two graphs of the class are isomorphic iff they agree on all k-variable formulas in fixed-point logic with counting.

Using Ehrenfeucht-Fraïssé games, this can be checked in polynomial time, $(O(n^k(\log n)))$. In the same time we can give a canonical description of the isomorphism type of any graph in the class. Thus every class of graphs with excluded minors admits the same general polynomial time canonization algorithm: we're isomorphic iff we agree on all formulas in C_k and in particular, you are isomorphic to me iff your C_k canonical description is equal to mine.

What We Know

- **Diagonalization**: more of the same resource gives us more:
  $\mathrm{DTIME}[n] \subsetneq \mathrm{DTIME}[n^2]$,
  same for DSPACE, NTIME, NSPACE, . . .

What We Know

- **Diagonalization**: more of the same resource gives us more:

  $\mathrm{DTIME}[n] \subsetneq \mathrm{DTIME}[n^2]$,

  same for DSPACE, NTIME, NSPACE, ...

- **Natural Complexity Classes have Natural Complete Problems**

  SAT for NP, CVAL for P, QSAT for PSPACE, ...

What We Know

- **Diagonalization**: more of the same resource gives us more:

  $\mathrm{DTIME}[n] \subsetneq \mathrm{DTIME}[n^2]$,

  same for DSPACE, NTIME, NSPACE, ...

- **Natural Complexity Classes have Natural Complete Problems**

  SAT for NP, CVAL for P, QSAT for PSPACE, ...

- **Major Missing Idea**: concept of work or conservation of energy in computation, i.e,

  in order to solve SAT or other hard problem we must do a certain amount of computational work.

Strong Lower Bounds on $\mathrm{FO}[t(n)]$ for small $t(n)$

- [Sipser]: strict first-order alternation hierarchy: FO.

Strong Lower Bounds on $\mathrm{FO}[t(n)]$ for small $t(n)$

- [Sipser]: strict first-order alternation hierarchy: FO.
- [Beame-Håstad]: hierarchy remains strict up to $\mathrm{FO}[\log n / \log \log n]$.

Strong Lower Bounds on $\mathrm{FO}[t(n)]$ for small $t(n)$

- [Sipser]: strict first-order alternation hierarchy: FO.
- [Beame-Håstad]: hierarchy remains strict up to $\mathrm{FO}[\log n / \log\log n]$.
- $\mathrm{NC}^1 \subseteq \mathrm{FO}[\log n / \log\log n]$ and this is tight.

Strong Lower Bounds on $\mathrm{FO}[t(n)]$ for small $t(n)$

- [Sipser]: strict first-order alternation hierarchy: FO.
- [Beame-Håstad]: hierarchy remains strict up to $\mathrm{FO}[\log n/\log\log n]$.
- $\mathrm{NC}^1 \subseteq \mathrm{FO}[\log n/\log\log n]$ and this is tight.
- Does REACH require $\mathrm{FO}[\log n]$? This would imply $\mathrm{NC}^1 \neq \mathrm{NL}$.

Does It Matter? How important is P $\neq$ NP?

- Much is known about approximation, e.g., some NP complete problems, e.g., Knapsack, Euclidean TSP, can be approximated as closely as we want, others, e.g., Clique, can't be.

Does It Matter? How important is P $\neq$ NP?

- Much is known about approximation, e.g., some NP complete problems, e.g., Knapsack, Euclidean TSP, can be approximated as closely as we want, others, e.g., Clique, can't be.
- We conjecture that SAT requires $\mathrm{DTIME}[\Omega(2^{\epsilon n})]$ for some $\epsilon > 0$, but no one has yet proved that it requires more than $\mathrm{DTIME}[n]$.

Does It Matter? How important is P $\neq$ NP?

- Much is known about approximation, e.g., some NP complete problems, e.g., Knapsack, Euclidean TSP, can be approximated as closely as we want, others, e.g., Clique, can't be.
- We conjecture that SAT requires $\mathrm{DTIME}[\Omega(2^{\epsilon n})]$ for some $\epsilon > 0$, but no one has yet proved that it requires more than $\mathrm{DTIME}[n]$.
- Basic trade-offs are not understood, e.g., trade-off between time and number of processors. Are any problems inherently sequential? How can we best use mulitcores?

Does It Matter? How important is P $\neq$ NP?

- Much is known about approximation, e.g., some NP complete problems, e.g., Knapsack, Euclidean TSP, can be approximated as closely as we want, others, e.g., Clique, can't be.
- We conjecture that SAT requires $\mathrm{DTIME}[\Omega(2^{\epsilon n})]$ for some $\epsilon > 0$, but no one has yet proved that it requires more than $\mathrm{DTIME}[n]$.
- Basic trade-offs are not understood, e.g., trade-off between time and number of processors. Are any problems inherently sequential? How can we best use mulitcores?
- **SAT solvers** are impressive new general purpose problem solvers, e.g., used in model checking, AI planning, code synthesis. How good are current SAT solvers? How much can they be improved?

Descriptive Complexity

Fact: For constructible $t(n)$, $\mathrm{FO}[t(n)] \;=\; \mathrm{CRAM}[t(n)]$

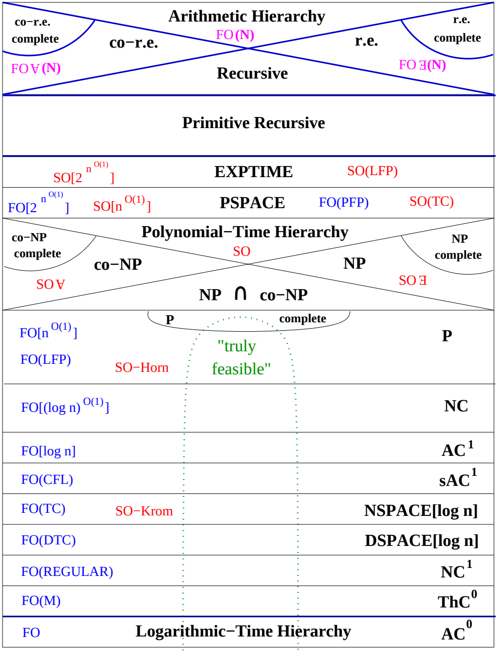
Fact: For $k = 1, 2, \ldots,$ $\mathrm{VAR}[k+1] \;=\; \mathrm{DSPACE}[n^k]$

The complexity of computing a query is closely tied to the complexity of describing the query.

$$\mathrm{P} = \mathrm{NP} \quad \Leftrightarrow \quad \mathrm{FO(LFP)} = \mathrm{SO}$$

$$\mathrm{ThC}^0 = \mathrm{NP} \quad \Leftrightarrow \quad \mathrm{FO(MAJ)} = \mathrm{SO}$$

$$\mathrm{P} = \mathrm{PSPACE} \quad \Leftrightarrow \quad \mathrm{FO(LFP)} = \mathrm{SO(TC)}$$


Arithmetic Hierarchy
co−r.e. complete
co−r.e.
FO (N)
r.e.
r.e. complete
FO ∀ (N)
Recursive
FO ∃(N)
Primitive Recursive
SO[2^n^O(1)]
EXPTIME
SO(LFP)
FO[2^n^O(1)]
SO[n^O(1)]
PSPACE
FO(PFP)
SO(TC)
Polynomial−Time Hierarchy
co−NP complete
co−NP
SO
NP
NP complete
SO ∀
SO ∃
NP ∩ co−NP
P complete
FO[n^O(1)]
"truly feasible"
P
FO(LFP)
SO−Horn
FO[(log n)^O(1)]
NC
FO[log n]
AC^1
FO(CFL)
sAC^1
FO(TC)
SO−Krom
NSPACE[log n]
FO(DTC)
DSPACE[log n]
FO(REGULAR)
NC^1
FO(M)
ThC^0
FO
Logarithmic−Time Hierarchy
AC^0