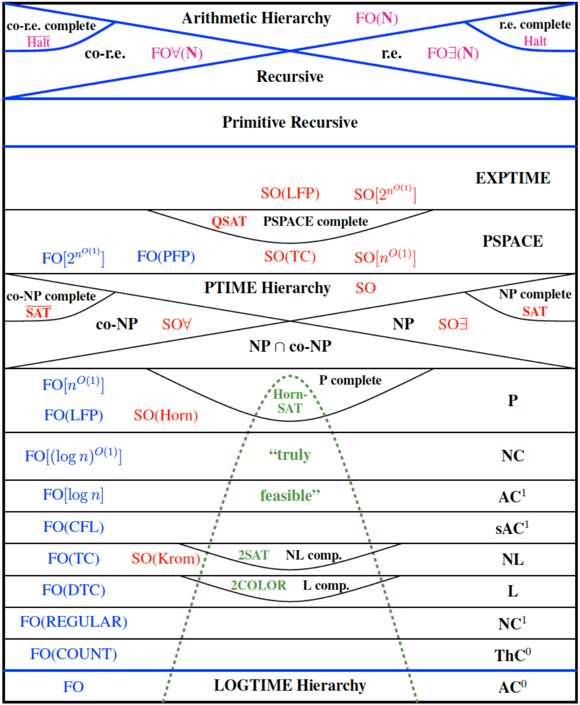
FO Isomorphism Theorems and Descriptive Complexity

Neil Immerman

UMasss, Amherst

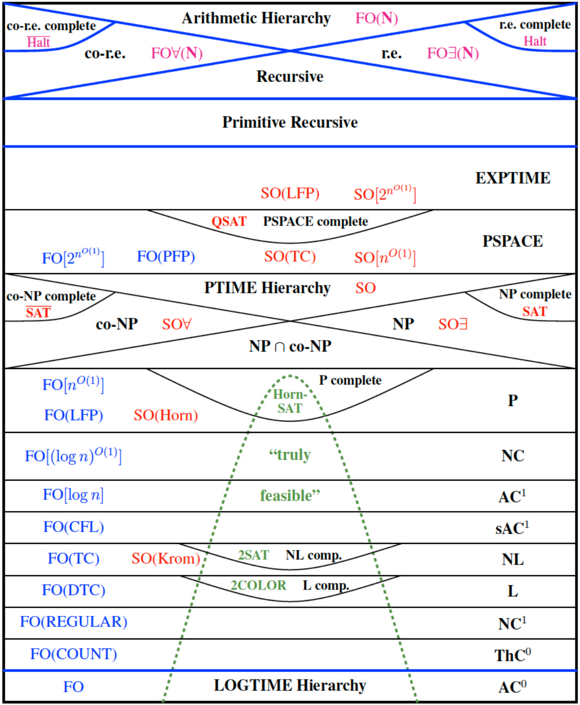
people.cs.umass.edu/~immerman

“truly feasible” is the informal set of problems we can solve exactly on all reasonably sized instances.



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

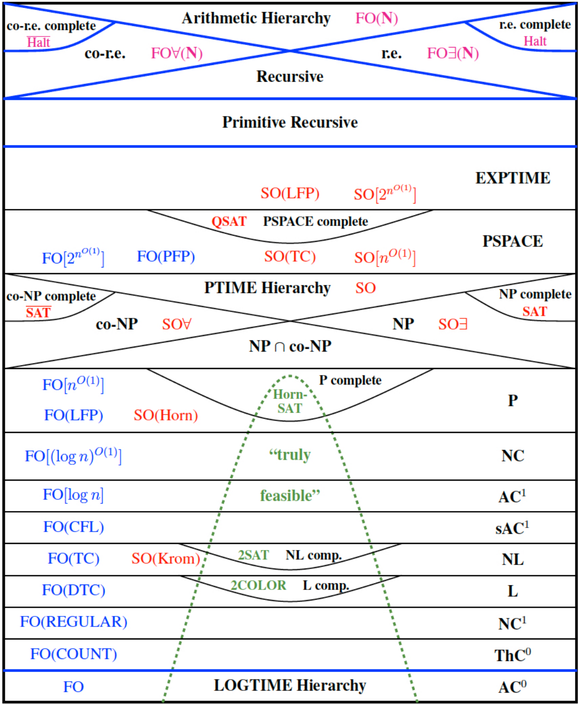
“truly feasible” is the informal set of problems we can solve exactly on all reasonably sized instances.



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

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

"truly feasible" is the informal set of problems we can solve exactly on all reasonably sized instances.



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

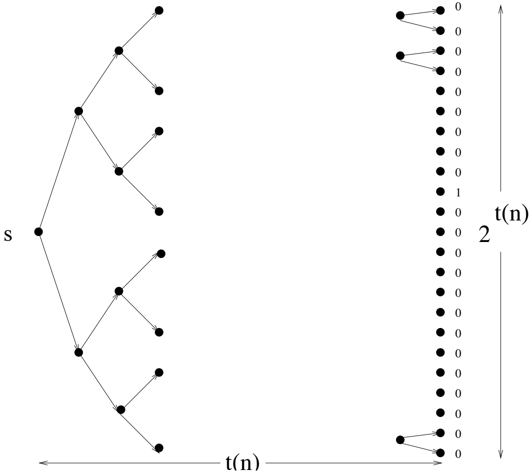
input w, $|w| = n$

N accepts w

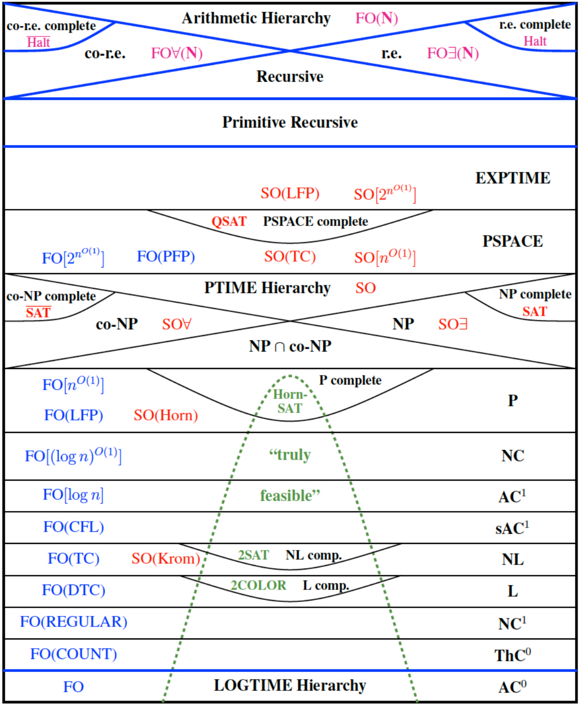
if at least

one of the $2^{t(n)}$

paths accepts.



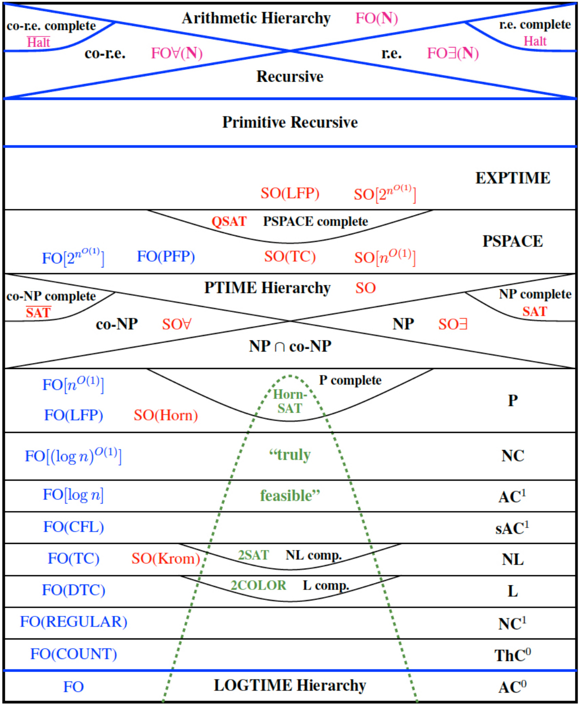
$$\mathrm{NP} = \bigcup_{k=1}^{\infty} \mathrm{NTIME}[n^k]$$



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

Many optimization problems we want to solve are NP complete.

SAT, TSP, 3-COLOR, CLIQUE, . . .

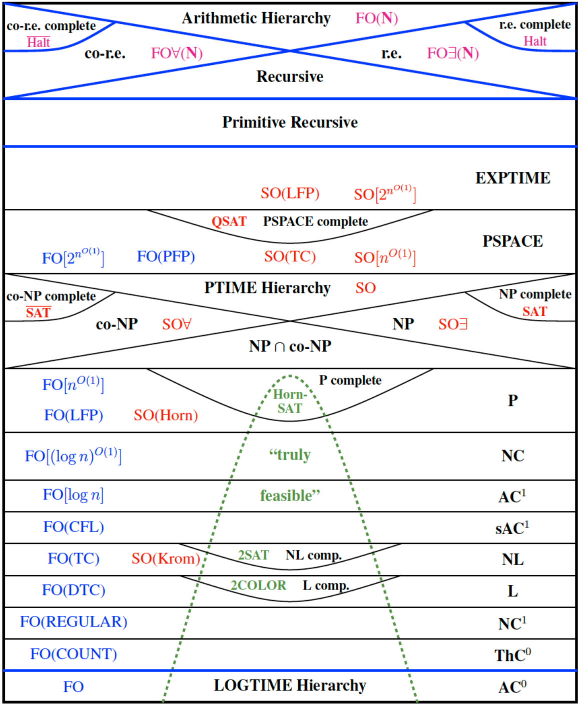


$$\mathrm{NP} = \bigcup_{k=1}^{\infty} \mathrm{NTIME}[n^k]$$

Many optimization problems we want to solve are NP complete.

SAT, TSP, 3-COLOR, CLIQUE, . . .

As descision problems, all NP complete problems are isomorphic.



Descriptive Complexity

$$\begin{array}{c}\textbf{Input}\\ q_1\ q_2\ \cdots\ q_n\end{array} \mapsto \boxed{\textbf{Computation}} \mapsto \begin{array}{c}\textbf{Answer}\\ a_1\ a_2\ \cdots\ a_i\ \cdots\ a_m\end{array}$$

Descriptive Complexity

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

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

Descriptive Complexity

$$\underset{q_1\ q_2\ \cdots\ q_n}{\textbf{Input}} \quad \mapsto \quad \boxed{\textbf{Computation}} \quad \mapsto \quad \underset{\substack{a_1\ a_2\ \cdots\ a_i\ \cdots\ a_m \\ \cdots\ S\ \cdots}}{\textbf{Answer}}$$

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 ?

Descriptive Complexity

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

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?

Descriptive Complexity

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

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.

Think of the Input as a Finite Logical Structure

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

$\Sigma_g = (E^2, s, t)$

Binary String $\qquad \mathcal{A}_w = (\{p_1, \ldots, p_8\}, \leq, S)$

$S = \{p_2, p_5, p_7, p_8\}$

$\Sigma_s = (S^1) \qquad w = 01001011$

First-Order Logic

input symbols:	from Σ
variables:	$x, y, z, \ldots$
boolean connectives:	$\wedge, \vee, \neg$
quantifiers:	$\forall, \exists$
numeric symbols:	$=, \leq, +, \times, min, max$

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

$$\beta \equiv \exists x \forall y (x \leq y \wedge S(x)) \in \mathcal{L}(\Sigma_s)$$

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

First-Order Logic

input symbols:	from Σ
variables:	$x, y, z, \ldots$
boolean connectives:	$\wedge, \vee, \neg$
quantifiers:	$\forall, \exists$
numeric symbols:	$=, \leq, +, \times, min, max$

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

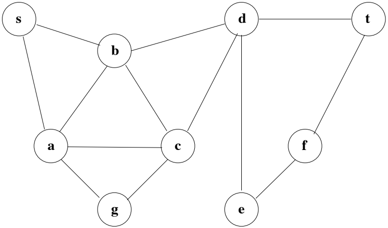
$$\beta \equiv \exists x \forall y (x \leq y \wedge S(x)) \in \mathcal{L}(\Sigma_s)$$

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

In this setting, with the structure of interest being the **finite input**, FO is a weak, low-level complexity class.

Second-Order Logic: FO plus Relation Variables

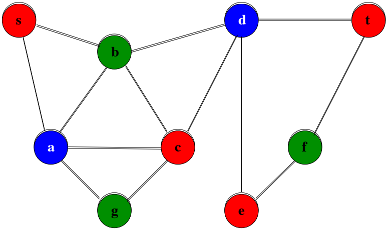
$$\Phi_{3\text{color}} \equiv \exists R^1 G^1 B^1 \forall x\, y\, ((R(x) \vee G(x) \vee B(x)) \wedge (E(x,y) \rightarrow$$
$$(\neg(R(x) \wedge R(y)) \wedge \neg(G(x) \wedge G(y)) \wedge \neg(B(x) \wedge B(y)))))$$

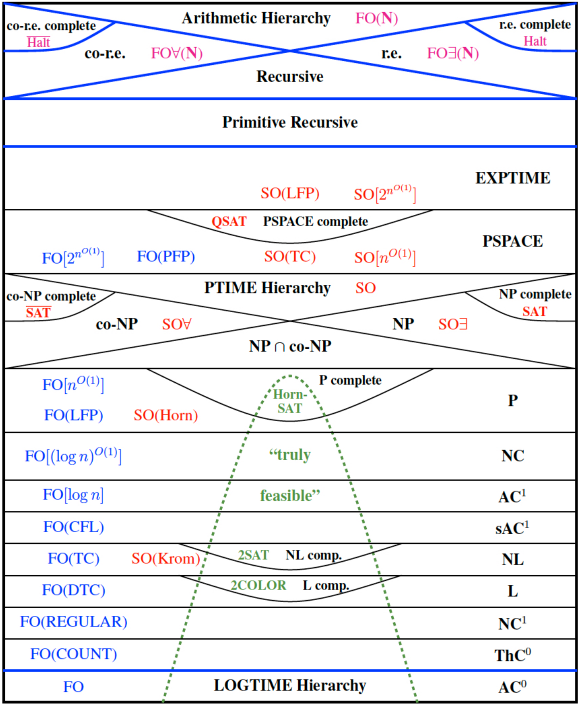


Second-Order Logic: FO plus Relation Variables

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

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





Arithmetic Hierarchy
FO(N)
co-r.e. complete
$\overline{\text{Halt}}$
r.e. complete
Halt
co-r.e.
FO∀(N)
r.e.
FO∃(N)
Recursive
Primitive Recursive
EXPTIME
SO(LFP)
$SO[2^{n^{O(1)}}]$
QSAT
PSPACE complete
PSPACE
$FO[2^{n^{O(1)}}]$
FO(PFP)
SO(TC)
$SO[n^{O(1)}]$
PTIME Hierarchy
SO
co-NP complete
$\overline{\text{SAT}}$
NP complete
SAT
co-NP
SO∀
NP
SO∃
NP ∩ co-NP
$FO[n^{O(1)}]$
P complete
Horn-SAT
P
FO(LFP)
SO(Horn)
$FO[(\log n)^{O(1)}]$
"truly
NC
$FO[\log n]$
feasible"
AC^1
FO(CFL)
sAC^1
FO(TC)
SO(Krom)
2SAT
NL comp.
NL
FO(DTC)
2COLOR
L comp.
L
FO(REGULAR)
NC^1
FO(COUNT)
ThC^0
FO
LOGTIME Hierarchy
AC^0

Addition is First-Order

$$Q_+ : \mathrm{STRUC}[\Sigma_{AB}] \to \mathrm{STRUC}[\Sigma_s]$$

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

Addition is First-Order

$$Q_+ : \mathrm{STRUC}[\Sigma_{AB}] \to \mathrm{STRUC}[\Sigma_s]$$

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

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

Addition is First-Order

$$Q_+ : \mathrm{STRUC}[\Sigma_{AB}] \to \mathrm{STRUC}[\Sigma_s]$$

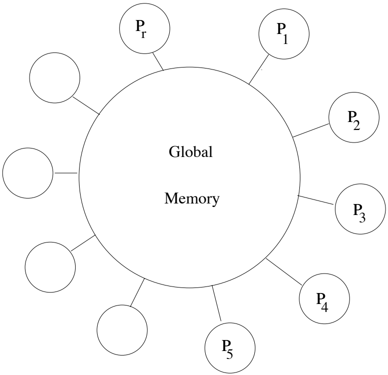
$$\begin{array}{llccccc} A & & a_1 & a_2 & \dots & a_{n-1} & a_n \\ B & + & b_1 & b_2 & \dots & b_{n-1} & b_n \\ \hline S & & s_1 & s_2 & \dots & s_{n-1} & s_n \end{array}$$

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

$$Q_+(i) \quad \equiv \quad A(i) \oplus B(i) \oplus C(i)$$

Parallel Machines:

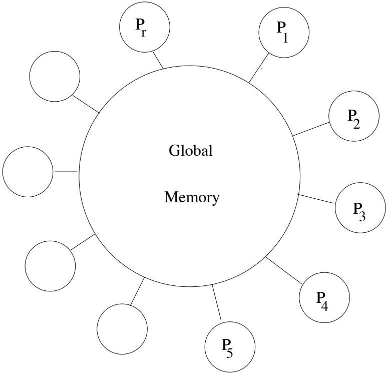
$$\mathrm{CRAM}[t(n)] \; = \mathrm{CRCW\text{-}PRAM\text{-}TIME}[t(n)]\text{-}\mathrm{HARD}[n^{O(1)}]$$



Parallel Machines: Quantifiers are Parallel

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

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

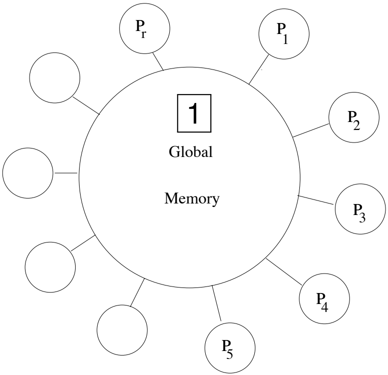


Parallel Machines: Quantifiers are Parallel

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

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

$\forall x(A(x)) \equiv$ **write**(1);

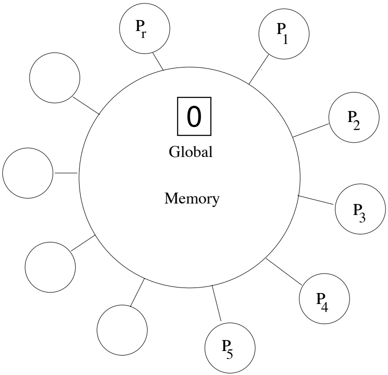


Parallel Machines: Quantifiers are Parallel

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

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

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



FO

=

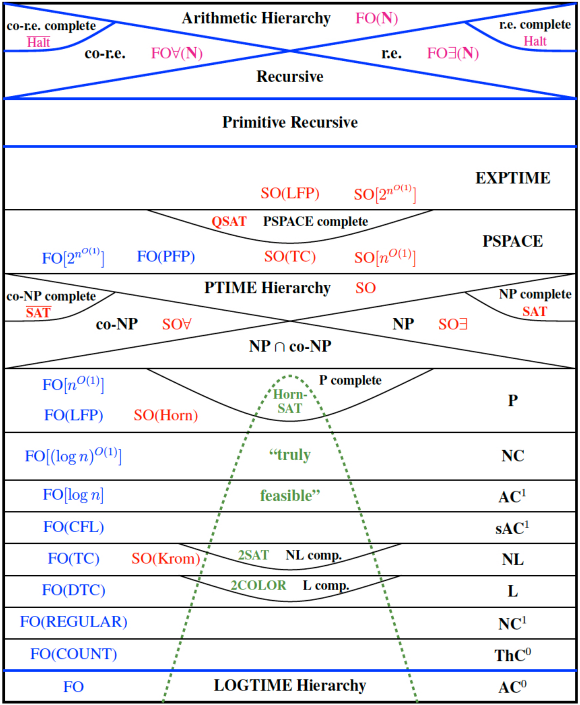
CRAM[1]

=

AC^0

=

Logarithmic-Time Hierarchy



Inductive Definitions and Least Fixed Point

$$\text{REACH} \;=\; \left\{G, s, t \;\middle|\; s \xrightarrow{\star} t\right\}$$

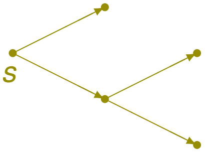
Inductive Definitions and Least Fixed Point

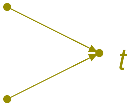
$\text{REACH} = \{G, s, t \mid s \xrightarrow{\star} t\}$ $\qquad$ $\text{REACH} \notin \mathrm{FO}$

Inductive Definitions and Least Fixed Point

$$E^\star(x,y) \quad \stackrel{\text{def}}{=} \quad x = y \ \lor\ E(x,y) \ \lor\ \exists z(E^\star(x,z) \land E^\star(z,y))$$

$$\text{REACH} \ = \ \left\{G, s, t \ \middle|\ s \stackrel{\star}{\to} t\right\} \qquad\qquad \text{REACH} \not\in \mathrm{FO}$$





Inductive Definitions and Least Fixed Point

$$E^\star(x,y) \stackrel{\text{def}}{=} x = y \lor E(x,y) \lor \exists z(E^\star(x,z) \land E^\star(z,y))$$

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

$$\text{REACH} = \{G,s,t \mid s \xrightarrow{\star} t\} \qquad \text{REACH} \notin \mathrm{FO}$$

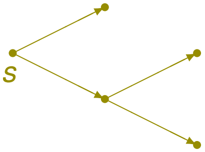
Inductive Definitions and Least Fixed Point

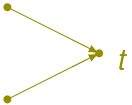
$$E^\star(x,y) \stackrel{\text{def}}{=} x = y \lor E(x,y) \lor \exists z(E^\star(x,z) \land E^\star(z,y))$$

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

$\varphi_{tc}^G : \mathrm{binRel}(G) \to \mathrm{binRel}(G)$ is a monotone operator

$$\mathrm{REACH} = \{G, s, t \mid s \stackrel{\star}{\to} t\} \qquad \mathrm{REACH} \notin \mathrm{FO}$$





Inductive Definitions and Least Fixed Point

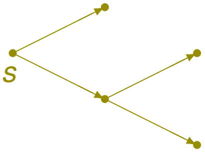
$$E^\star(x,y) \stackrel{\text{def}}{=} x = y \ \vee\ E(x,y) \ \vee\ \exists z(E^\star(x,z) \wedge E^\star(z,y))$$

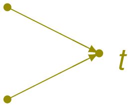
$$\varphi_{tc}(R,x,y) \equiv x = y \ \vee\ E(x,y) \ \vee\ \exists z(R(x,z) \wedge R(z,y))$$

$\varphi_{tc}^G : \text{binRel}(G) \to \text{binRel}(G)$ is a monotone operator

$$E^\star = (\mathrm{LFP}\varphi_{tc})$$

$$\text{REACH} = \{G, s, t \mid s \xrightarrow{\star} t\} \qquad \text{REACH} \notin \mathrm{FO}$$





Inductive Definitions and Least Fixed Point

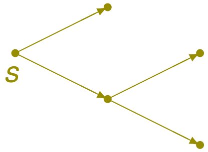
$$E^\star(x,y) \quad \stackrel{\text{def}}{=} \quad x = y \ \vee \ E(x,y) \ \vee \ \exists z(E^\star(x,z) \wedge E^\star(z,y))$$

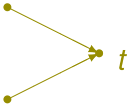
$$\varphi_{tc}(R,x,y) \quad \equiv \quad x = y \ \vee \ E(x,y) \ \vee \ \exists z(R(x,z) \wedge R(z,y))$$

$$\varphi_{tc}^G : \text{binRel}(G) \quad \to \quad \text{binRel}(G) \qquad \text{is a monotone operator}$$

$$G \in \text{REACH} \ \Leftrightarrow \ G \models (\mathrm{LFP}\varphi_{tc})(s,t) \qquad E^\star \ = \ (\mathrm{LFP}\varphi_{tc})$$

$$\text{REACH} \ = \ \left\{G, s, t \ \middle| \ s \stackrel{\star}{\to} t\right\} \qquad \text{REACH} \not\in \mathrm{FO}$$





Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Let $I^0 \stackrel{\text{def}}{=} \emptyset; \quad I^{r+1} \stackrel{\text{def}}{=} \varphi(I^r)$

Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Let $I^0 \stackrel{\text{def}}{=} \emptyset$; $\quad I^{r+1} \stackrel{\text{def}}{=} \varphi(I^r)$ $\quad$ Thus, $\emptyset = I^0 \subseteq I^1 \subseteq \cdots \subseteq I^t$.

Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Let $I^0 \stackrel{\text{def}}{=} \emptyset$; $\quad I^{r+1} \stackrel{\text{def}}{=} \varphi(I^r)$ $\quad$ Thus, $\emptyset = I^0 \subseteq I^1 \subseteq \cdots \subseteq I^t$.

Let t be min such that $I^t = I^{t+1}$. Note that $t \leq n^k$ where $n = |V^G|$.

Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Let $I^0 \stackrel{\text{def}}{=} \emptyset; \quad I^{r+1} \stackrel{\text{def}}{=} \varphi(I^r) \quad$ Thus, $\emptyset = I^0 \subseteq I^1 \subseteq \cdots \subseteq I^t$.

Let t be min such that $I^t = I^{t+1}$. Note that $t \leq n^k$ where $n = |V^G|$. $\quad \varphi(I^t) = I^t, \quad$ so I^t is a fixed point of φ.

Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Let $I^0 \stackrel{\text{def}}{=} \emptyset$; $\quad I^{r+1} \stackrel{\text{def}}{=} \varphi(I^r) \quad$ Thus, $\emptyset = I^0 \subseteq I^1 \subseteq \cdots \subseteq I^t$.

Let t be min such that $I^t = I^{t+1}$. Note that $t \leq n^k$ where $n = |V^G|$. $\quad \varphi(I^t) = I^t$, $\quad$ so I^t is a fixed point of φ.

Suppose $\varphi(F) = F$.

Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Let $I^0 \stackrel{\text{def}}{=} \emptyset; \quad I^{r+1} \stackrel{\text{def}}{=} \varphi(I^r)$ $\quad$ Thus, $\emptyset = I^0 \subseteq I^1 \subseteq \cdots \subseteq I^t$.

Let t be min such that $I^t = I^{t+1}$. Note that $t \leq n^k$ where $n = |V^G|$. $\quad \varphi(I^t) = I^t$, $\quad$ so I^t is a fixed point of φ.

Suppose $\varphi(F) = F$. $\quad$ By induction on r, for all r, $I^r \subseteq F$.

Tarski-Knaster Theorem

Thm. If $\varphi : \text{Rel}^k(G) \to \text{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Let $I^0 \stackrel{\text{def}}{=} \emptyset$; $\quad I^{r+1} \stackrel{\text{def}}{=} \varphi(I^r) \quad$ Thus, $\emptyset = I^0 \subseteq I^1 \subseteq \cdots \subseteq I^t$.

Let t be min such that $I^t = I^{t+1}$. Note that $t \leq n^k$ where $n = |V^G|$. $\quad \varphi(I^t) = I^t$, $\quad$ so I^t is a fixed point of φ.

Suppose $\varphi(F) = F$. $\quad$ By induction on r, for all r, $I^r \subseteq F$.

base case: $\quad I^0 = \emptyset \subseteq F$.

Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Let $I^0 \stackrel{\text{def}}{=} \emptyset$; $\quad I^{r+1} \stackrel{\text{def}}{=} \varphi(I^r)$ $\quad$ Thus, $\emptyset = I^0 \subseteq I^1 \subseteq \cdots \subseteq I^t$.

Let t be min such that $I^t = I^{t+1}$. Note that $t \leq n^k$ where $n = |V^G|$. $\quad \varphi(I^t) = I^t$, $\quad$ so I^t is a fixed point of φ.

Suppose $\varphi(F) = F$. $\quad$ By induction on r, for all r, $I^r \subseteq F$.

base case: $\quad I^0 = \emptyset \subseteq F$.

inductive case: $\quad$ Assume $I^j \subseteq F$

Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Let $I^0 \stackrel{\text{def}}{=} \emptyset$; $\quad I^{r+1} \stackrel{\text{def}}{=} \varphi(I^r)$ $\quad$ Thus, $\emptyset = I^0 \subseteq I^1 \subseteq \cdots \subseteq I^t$.

Let t be min such that $I^t = I^{t+1}$. Note that $t \leq n^k$ where $n = |V^G|$. $\quad \varphi(I^t) = I^t$, $\quad$ so I^t is a fixed point of φ.

Suppose $\varphi(F) = F$. $\quad$ By induction on r, for all r, $I^r \subseteq F$.

base case: $\quad I^0 = \emptyset \subseteq F$.

inductive case: $\quad$ Assume $I^j \subseteq F$

By monotonicity, $\quad \varphi(I^j) \subseteq \varphi(F)$, i.e., $\quad I^{j+1} \subseteq F$.

Tarski-Knaster Theorem

Thm. If $\varphi : \mathrm{Rel}^k(G) \to \mathrm{Rel}^k(G)$ is monotone, then $\mathrm{LFP}(\varphi)$ exists and can be computed in P.

proof: Monotone means, for all $R \subseteq S$, $\quad \varphi(R) \subseteq \varphi(S)$.

Let $I^0 \stackrel{\mathrm{def}}{=} \emptyset$; $\quad I^{r+1} \stackrel{\mathrm{def}}{=} \varphi(I^r)$ $\quad$ Thus, $\emptyset = I^0 \subseteq I^1 \subseteq \cdots \subseteq I^t$.

Let t be min such that $I^t = I^{t+1}$. Note that $t \leq n^k$ where $n = |V^G|$. $\quad \varphi(I^t) = I^t$, $\quad$ so I^t is a fixed point of φ.

Suppose $\varphi(F) = F$. $\quad$ By induction on r, for all r, $I^r \subseteq F$.

base case: $\quad I^0 = \emptyset \subseteq F$.

inductive case: $\quad$ Assume $I^j \subseteq F$

By monotonicity, $\quad \varphi(I^j) \subseteq \varphi(F)$, i.e., $\quad I^{j+1} \subseteq F$.

Thus $I^t \subseteq F$ $\quad$ and $\quad I^t = \mathrm{LFP}(\varphi)$. $\square$

Inductive Definition of Transitive Closure

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

Inductive Definition of Transitive Closure

$$\begin{aligned}
\varphi_{tc}(R, x, y) &\equiv x = y \lor E(x, y) \lor \exists z(R(x, z) \land R(z, y)) \\
I^1 = \varphi_{tc}^G(\emptyset) &= \left\{(a, b) \in V^G \times V^G \;\middle|\; \mathrm{dist}(a, b) \leq 1\right\}
\end{aligned}$$

Inductive Definition of Transitive Closure

$$\begin{array}{rcl} \varphi_{tc}(R,x,y) & \equiv & x = y \ \vee\ E(x,y) \ \vee\ \exists z(R(x,z) \wedge R(z,y)) \\ I^1 = \varphi_{tc}^G(\emptyset) & = & \big\{(a,b) \in V^G \times V^G \ \big|\ \mathrm{dist}(a,b) \leq 1\big\} \\ I^2 = (\varphi_{tc}^G)^2(\emptyset) & = & \big\{(a,b) \in V^G \times V^G \ \big|\ \mathrm{dist}(a,b) \leq 2\big\} \end{array}$$

Inductive Definition of Transitive Closure

$$
\begin{aligned}
\varphi_{tc}(R,x,y) \quad &\equiv \quad x = y \ \vee \ E(x,y) \ \vee \ \exists z (R(x,z) \wedge R(z,y)) \\
I^1 = \varphi_{tc}^G(\emptyset) \quad &= \quad \big\{(a,b) \in V^G \times V^G \ \big| \ \mathrm{dist}(a,b) \leq 1\big\} \\
I^2 = (\varphi_{tc}^G)^2(\emptyset) \quad &= \quad \big\{(a,b) \in V^G \times V^G \ \big| \ \mathrm{dist}(a,b) \leq 2\big\} \\
I^3 = (\varphi_{tc}^G)^3(\emptyset) \quad &= \quad \big\{(a,b) \in V^G \times V^G \ \big| \ \mathrm{dist}(a,b) \leq 4\big\}
\end{aligned}
$$

Inductive Definition of Transitive Closure

$$
\begin{array}{ccc}
\varphi_{tc}(R,x,y) & \equiv & x = y \ \vee \ E(x,y) \ \vee \ \exists z(R(x,z) \wedge R(z,y)) \\
I^1 = \varphi_{tc}^G(\emptyset) & = & \big\{(a,b) \in V^G \times V^G \ \big| \ \mathrm{dist}(a,b) \leq 1\big\} \\
I^2 = (\varphi_{tc}^G)^2(\emptyset) & = & \big\{(a,b) \in V^G \times V^G \ \big| \ \mathrm{dist}(a,b) \leq 2\big\} \\
I^3 = (\varphi_{tc}^G)^3(\emptyset) & = & \big\{(a,b) \in V^G \times V^G \ \big| \ \mathrm{dist}(a,b) \leq 4\big\} \\
\vdots & = & \vdots \qquad\qquad\qquad \vdots \\
I^r = (\varphi_{tc}^G)^r(\emptyset) & = & \big\{(a,b) \in V^G \times V^G \ \big| \ \mathrm{dist}(a,b) \leq 2^{r-1}\big\} \\
\vdots & = & \vdots \qquad\qquad\qquad \vdots
\end{array}
$$

Inductive Definition of Transitive Closure

$$
\begin{array}{rcc}
\varphi_{tc}(R,x,y) & \equiv & x=y \ \vee \ E(x,y) \ \vee \ \exists z(R(x,z)\wedge R(z,y)) \\
I^1 = \varphi_{tc}^G(\emptyset) & = & \{(a,b)\in V^G\times V^G \mid \mathrm{dist}(a,b)\le 1\} \\
I^2 = (\varphi_{tc}^G)^2(\emptyset) & = & \{(a,b)\in V^G\times V^G \mid \mathrm{dist}(a,b)\le 2\} \\
I^3 = (\varphi_{tc}^G)^3(\emptyset) & = & \{(a,b)\in V^G\times V^G \mid \mathrm{dist}(a,b)\le 4\} \\
\vdots & = & \vdots \qquad\qquad\qquad \vdots \\
I^r = (\varphi_{tc}^G)^r(\emptyset) & = & \{(a,b)\in V^G\times V^G \mid \mathrm{dist}(a,b)\le 2^{r-1}\} \\
\vdots & = & \vdots \qquad\qquad\qquad \vdots \\
(\varphi_{tc}^G)^{\lceil 1+\log n\rceil}(\emptyset) & = & \{(a,b)\in V^G\times V^G \mid \mathrm{dist}(a,b)\le n\}
\end{array}
$$

Inductive Definition of Transitive Closure

$$
\begin{array}{rcl}
\varphi_{tc}(R,x,y) & \equiv & x=y \ \vee \ E(x,y) \ \vee \ \exists z(R(x,z)\wedge R(z,y)) \\
I^1 = \varphi_{tc}^G(\emptyset) & = & \left\{(a,b)\in V^G\times V^G \;\middle|\; \mathrm{dist}(a,b)\le 1\right\} \\
I^2 = (\varphi_{tc}^G)^2(\emptyset) & = & \left\{(a,b)\in V^G\times V^G \;\middle|\; \mathrm{dist}(a,b)\le 2\right\} \\
I^3 = (\varphi_{tc}^G)^3(\emptyset) & = & \left\{(a,b)\in V^G\times V^G \;\middle|\; \mathrm{dist}(a,b)\le 4\right\} \\
\vdots & = & \qquad\vdots \qquad\qquad\qquad \vdots \\
I^r = (\varphi_{tc}^G)^r(\emptyset) & = & \left\{(a,b)\in V^G\times V^G \;\middle|\; \mathrm{dist}(a,b)\le 2^{r-1}\right\} \\
\vdots & = & \qquad\vdots \qquad\qquad\qquad \vdots \\
(\varphi_{tc}^G)^{\lceil 1+\log n\rceil}(\emptyset) & = & \left\{(a,b)\in V^G\times V^G \;\middle|\; \mathrm{dist}(a,b)\le n\right\} \\
\mathrm{LFP}(\varphi_{tc}) & = & \varphi_{tc}^{\lceil 1+\log n\rceil}(\emptyset); \qquad \text{REACH}\in \mathrm{IND}[\log n]
\end{array}
$$

Inductive Definition of Transitive Closure

$$
\begin{array}{rcl}
\varphi_{tc}(R,x,y) & \equiv & x = y \ \vee \ E(x,y) \ \vee \ \exists z(R(x,z) \wedge R(z,y)) \\
I^1 = \varphi_{tc}^G(\emptyset) & = & \left\{(a,b) \in V^G \times V^G \ \middle|\ \mathrm{dist}(a,b) \le 1\right\} \\
I^2 = (\varphi_{tc}^G)^2(\emptyset) & = & \left\{(a,b) \in V^G \times V^G \ \middle|\ \mathrm{dist}(a,b) \le 2\right\} \\
I^3 = (\varphi_{tc}^G)^3(\emptyset) & = & \left\{(a,b) \in V^G \times V^G \ \middle|\ \mathrm{dist}(a,b) \le 4\right\} \\
\vdots & = & \vdots \qquad\qquad \vdots \\
I^r = (\varphi_{tc}^G)^r(\emptyset) & = & \left\{(a,b) \in V^G \times V^G \ \middle|\ \mathrm{dist}(a,b) \le 2^{r-1}\right\} \\
\vdots & = & \vdots \qquad\qquad \vdots \\
(\varphi_{tc}^G)^{\lceil 1+\log n \rceil}(\emptyset) & = & \left\{(a,b) \in V^G \times V^G \ \middle|\ \mathrm{dist}(a,b) \le n\right\} \\
\mathrm{LFP}(\varphi_{tc}) & = & \varphi_{tc}^{\lceil 1+\log n \rceil}(\emptyset); \qquad \text{REACH} \in \mathrm{IND}[\log n]
\end{array}
$$

Next we will show that $\mathrm{IND}[t(n)] \ = \ \mathrm{FO}[t(n)]$.

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

1. Dummy universal quantification for base case:

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

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

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:

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

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

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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

parallel time = inductive depth = QB iteration

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

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

parallel time = inductive depth = QB iteration

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

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

proof idea: $\mathrm{CRAM}[t(n)] \supseteq \mathrm{FO}[t(n)]$: For QB with k variables, keep in memory current value of formula on all possible assignments, using n^k bits of global memory.

parallel time = inductive depth = QB iteration

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

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

proof idea: $\mathrm{CRAM}[t(n)] \supseteq \mathrm{FO}[t(n)]$: For QB with k variables, keep in memory current value of formula on all possible assignments, using n^k bits of global memory.
Simulate each next quantifier in constant parallel time.

parallel time = inductive depth = QB iteration

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

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

proof idea: $\mathrm{CRAM}[t(n)] \supseteq \mathrm{FO}[t(n)]$: For QB with k variables, keep in memory current value of formula on all possible assignments, using n^k bits of global memory.
Simulate each next quantifier in constant parallel time.

$\mathrm{CRAM}[t(n)] \subseteq \mathrm{FO}[t(n)]$: Inductively define new state of every bit of every register of every processor in terms of this global state at the previous time step. $\square$

parallel time = inductive depth = QB iteration

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

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

proof idea: $\mathrm{CRAM}[t(n)] \supseteq \mathrm{FO}[t(n)]$: For QB with k variables, keep in memory current value of formula on all possible assignments, using n^k bits of global memory.
Simulate each next quantifier in constant parallel time.

$\mathrm{CRAM}[t(n)] \subseteq \mathrm{FO}[t(n)]$: Inductively define new state of every bit of every register of every processor in terms of this global state at the previous time step. $\square$

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

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

For $t(n)$ poly bdd,

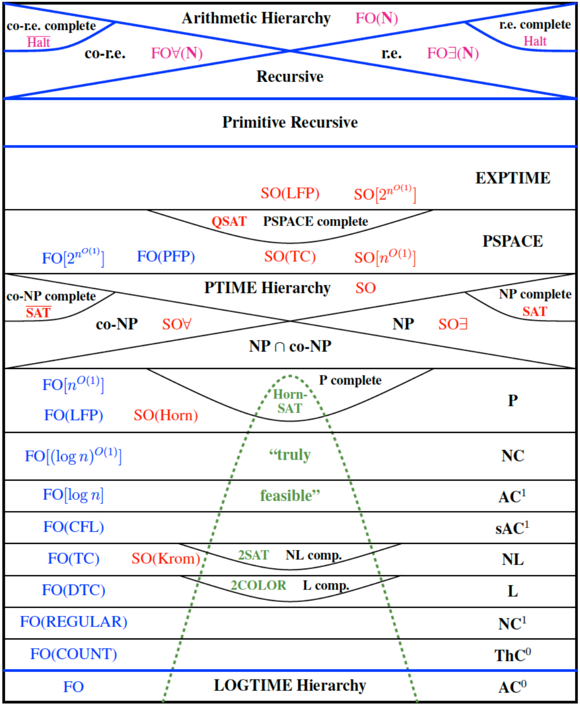
$\mathrm{CRAM}[t(n)]$

$=$

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

$=$

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



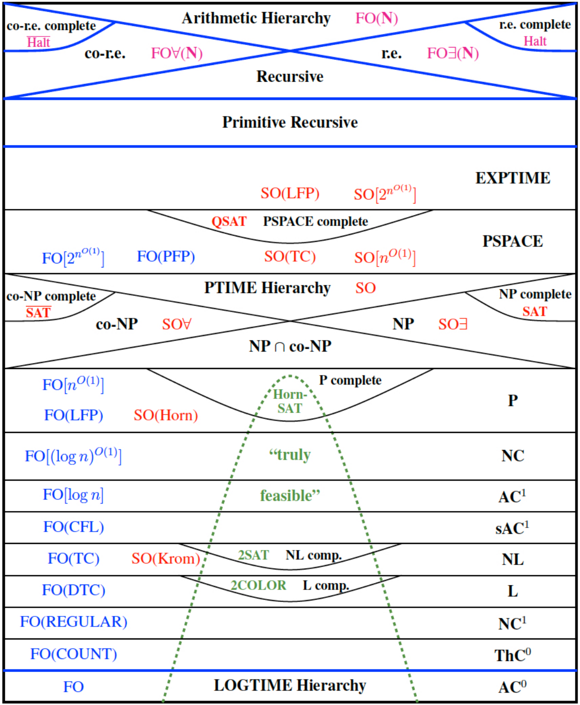
Remember that

for all $t(n)$,

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

$$=$$

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



Number of Variables Determines Amount of Hardware

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

Number of Variables Determines Amount of Hardware

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

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

Number of Variables Determines Amount of Hardware

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

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

The proof is just a more detailed look at $\mathrm{CRAM}[t(n)] = \mathrm{FO}[t(n)]$.

Number of Variables Determines Amount of Hardware

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

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

The proof is just a more detailed look at $\mathrm{CRAM}[t(n)] = \mathrm{FO}[t(n)]$.

A bounded number, k, of variables, is $k \log n$ bits and corresponds to n^k gates, i.e., polynomially much hardware.

Number of Variables Determines Amount of Hardware

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

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

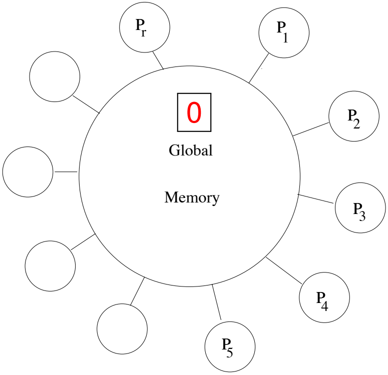
The proof is just a more detailed look at $\mathrm{CRAM}[t(n)] = \mathrm{FO}[t(n)]$.

A bounded number, k, of variables, is $k \log n$ bits and corresponds to n^k gates, i.e., polynomially much hardware.

A second-order variable of arity r is n^r bits, corresponding to 2^{n^r} gates.

SO: Parallel Machines with Exponential Hardware

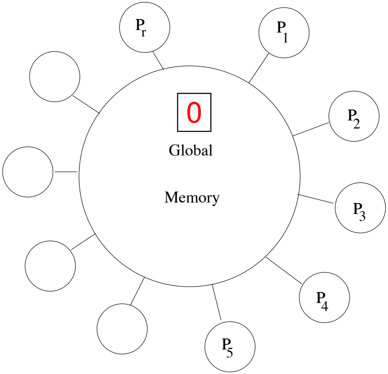
Given φ with n variables and m clauses, is $\varphi \in$ 3-SAT?



SO: Parallel Machines with Exponential Hardware

Given φ with n variables and m clauses, is $\varphi \in$ 3-SAT?

With $r = m2^n$ processors, recognize 3-SAT in constant time!

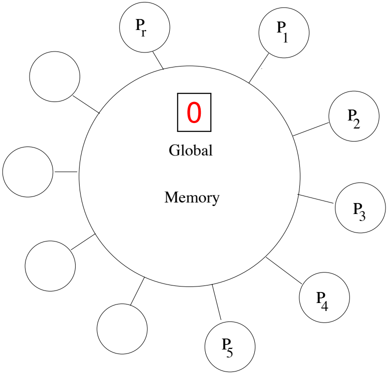


SO: Parallel Machines with Exponential Hardware

Given φ with n variables and m clauses, is $\varphi \in$ 3-SAT?

With $r = m2^n$ processors, recognize 3-SAT in constant time!

Let S be the first n bits of our processor number.

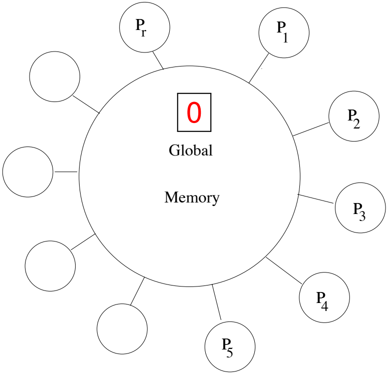


SO: Parallel Machines with Exponential Hardware

Given φ with n variables and m clauses, is $\varphi \in$ 3-SAT?

With $r = m2^n$ processors, recognize 3-SAT in constant time!

Let S be the first n bits of our processor number.
If processors $S1, \ldots Sm$ notice that truth assignment S makes all m clauses of φ true, then $\varphi \in$ 3-SAT,

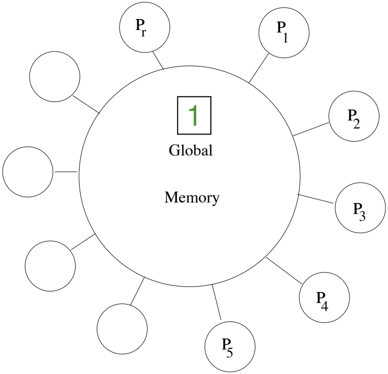


SO: Parallel Machines with Exponential Hardware

Given φ with n variables and m clauses, is $\varphi \in$ 3-SAT?

With $r = m2^n$ processors, recognize 3-SAT in constant time!

Let S be the first n bits of our processor number.
If processors $S1, \ldots Sm$ notice that truth assignment S makes all m clauses of φ true, then $\varphi \in$ 3-SAT, so $S1$ writes a 1.



SO: Parallel Machines with Exponential Hardware

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

SO: Parallel Machines with Exponential Hardware

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

proof: $\mathrm{SO}[t(n)]$ is like $\mathrm{FO}[t(n)]$ but using a quantifier block containing both first-order and second-order quantifiers.
The proof is similar to $\mathrm{FO}[t(n)] = \mathrm{CRAM}[t(n)]$. $\square$

SO: Parallel Machines with Exponential Hardware

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

proof: $\mathrm{SO}[t(n)]$ is like $\mathrm{FO}[t(n)]$ but using a quantifier block containing both first-order and second-order quantifiers.
The proof is similar to $\mathrm{FO}[t(n)] = \mathrm{CRAM}[t(n)]$. □

Cor.

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

SO: Parallel Machines with Exponential Hardware

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

proof: $\mathrm{SO}[t(n)]$ is like $\mathrm{FO}[t(n)]$ but using a quantifier block containing both first-order and second-order quantifiers.
The proof is similar to $\mathrm{FO}[t(n)] = \mathrm{CRAM}[t(n)]$. $\square$

Cor.

$$\begin{array}{ccccc} \mathrm{SO} & = & \text{PTIME Hierarchy} & = & \mathrm{CRAM}[1]\text{-}\mathrm{HARD}[2^{n^{O(1)}}] \\ \mathrm{SO}[n^{O(1)}] & = & \mathrm{PSPACE} & = & \mathrm{CRAM}[n^{O(1)}]\text{-}\mathrm{HARD}[2^{n^{O(1)}}] \end{array}$$

SO: Parallel Machines with Exponential Hardware

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

proof: $\mathrm{SO}[t(n)]$ is like $\mathrm{FO}[t(n)]$ but using a quantifier block containing both first-order and second-order quantifiers. The proof is similar to $\mathrm{FO}[t(n)] = \mathrm{CRAM}[t(n)]$. $\square$

Cor.

$$
\begin{array}{ccccc}
\mathrm{SO} & = & \text{PTIME Hierarchy} & = & \mathrm{CRAM}[1]\text{-}\mathrm{HARD}[2^{n^{O(1)}}] \\
\mathrm{SO}[n^{O(1)}] & = & \mathrm{PSPACE} & = & \mathrm{CRAM}[n^{O(1)}]\text{-}\mathrm{HARD}[2^{n^{O(1)}}] \\
\mathrm{SO}[2^{n^{O(1)}}] & = & \mathrm{EXPTIME} & = & \mathrm{CRAM}[2^{n^{O(1)}}]\text{-}\mathrm{HARD}[2^{n^{O(1)}}]
\end{array}
$$

Parallel Time versus Amount of Hardware

$$
\begin{aligned}
\mathrm{PSPACE} &= \mathrm{FO}[2^{n^{O(1)}}] &&= \mathrm{CRAM}[2^{n^{O(1)}}]\text{-}\mathrm{HARD}[\mathrm{n}^{O(1)}] \\
&= \mathrm{SO}[n^{O(1)}] &&= \mathrm{CRAM}[n^{O(1)}]\text{-}\mathrm{HARD}[2^{\mathrm{n}^{O(1)}}]
\end{aligned}
$$

Parallel Time versus Amount of Hardware

$$
\begin{aligned}
\text{PSPACE} &= \text{FO}[2^{n^{O(1)}}] &= \text{CRAM}[2^{n^{O(1)}}]\text{-HARD}[\mathrm{n}^{\mathrm{O}(1)}] \\
&= \text{SO}[n^{O(1)}] &= \text{CRAM}[n^{O(1)}]\text{-HARD}[2^{\mathrm{n}^{\mathrm{O}(1)}}]
\end{aligned}
$$

- We would love to understand this tradeoff.

Parallel Time versus Amount of Hardware

$$\begin{aligned}\mathrm{PSPACE} &= \mathrm{FO}[2^{n^{O(1)}}] &= \mathrm{CRAM}[2^{n^{O(1)}}]\text{-}\mathrm{HARD}[\mathrm{n}^{\mathrm{O}(1)}]\\ &= \mathrm{SO}[n^{O(1)}] &= \mathrm{CRAM}[n^{O(1)}]\text{-}\mathrm{HARD}[2^{\mathrm{n}^{\mathrm{O}(1)}}]\end{aligned}$$

- We would love to understand this tradeoff.
- Is there such a thing as an inherently sequential problem?, i.e., is $\mathrm{NC} \neq \mathrm{P}$?

Parallel Time versus Amount of Hardware

$$\begin{aligned}
\mathrm{PSPACE} &= \mathrm{FO}[2^{n^{O(1)}}] &= \mathrm{CRAM}[2^{n^{O(1)}}]\text{-}\mathrm{HARD}[\mathrm{n}^{O(1)}] \\
&= \mathrm{SO}[n^{O(1)}] &= \mathrm{CRAM}[n^{O(1)}]\text{-}\mathrm{HARD}[2^{\mathrm{n}^{O(1)}}]
\end{aligned}$$

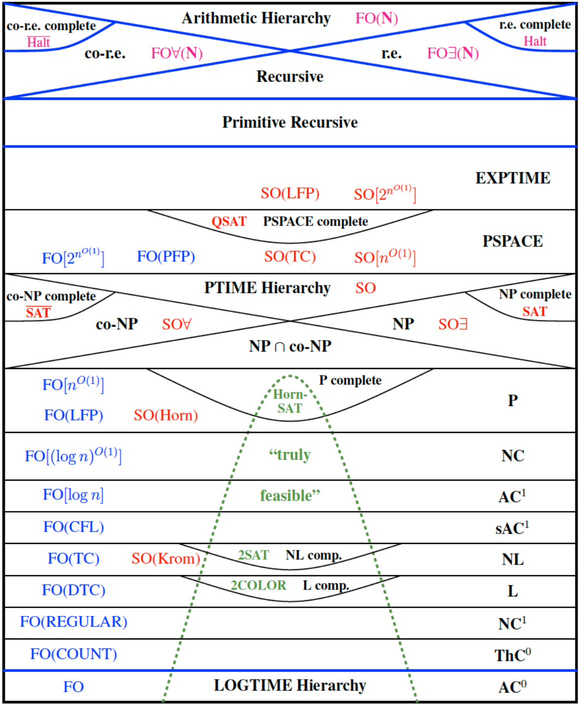
- We would love to understand this tradeoff.
- Is there such a thing as an inherently sequential problem?, i.e., is $\mathrm{NC} \neq \mathrm{P}$?
- Same tradeoff as number of variables vs. number of iterations of a quantifier block.

$\mathrm{SO}[t(n)]$

$=$

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

$\mathrm{HARD}\text{-}[2^{n^{O(1)}}]$



Reductions

- [C71] SAT is NP complete via ptime Turing reductions.

Reductions

- [C71] SAT is NP complete via ptime Turing reductions.
- [K72] Many important problems are NP complete, via $\leq_{\mathrm{p}}$.

Reductions

- [C71] SAT is NP complete via ptime Turing reductions.
- [K72] Many important problems are NP complete, via $\leq_{\mathrm{p}}$.
- [J73] … stay complete via **logspace** reductions, $\leq_{\mathrm{log}}$.

Reductions

- [C71] SAT is NP complete via ptime Turing reductions.
- [K72] Many important problems are NP complete, via $\leq_{\text{p}}$.
- [J73] … stay complete via **logspace** reductions, $\leq_{\text{log}}$.
- [HIM78] … stay complete via **one-way logspace**, reductions, $\leq_{\text{1-log}}$.

Reductions

- [C71] SAT is NP complete via ptime Turing reductions.
- [K72] Many important problems are NP complete, via $\leq_{\mathrm{p}}$.
- [J73] ...stay complete via **logspace** reductions, $\leq_{\mathrm{log}}$.
- [HIM78] ...stay complete via **one-way logspace**, reductions, $\leq_{\mathrm{1\text{-}log}}$.
- [I80] ...stay complete **first-order** reductions.

Reductions

- [C71] SAT is NP complete via ptime Turing reductions.
- [K72] Many important problems are NP complete, via $\leq_{\mathrm{p}}$.
- [J73] ... stay complete via **logspace** reductions, $\leq_{\mathrm{log}}$.
- [HIM78] ... stay complete via **one-way logspace**, reductions, $\leq_{\text{1-log}}$.
- [I80] ... stay complete **first-order** reductions.
- [V82] ... stay complete via **projections**. (Non-uniform reductions where each bit of the output depends on at most one bit of the input).

Reductions

- [C71] SAT is NP complete via ptime Turing reductions.
- [K72] Many important problems are NP complete, via $\leq_{\mathrm{p}}$.
- [J73] . . . stay complete via **logspace** reductions, $\leq_{\mathrm{log}}$.
- [HIM78] . . . stay complete via **one-way logspace**, reductions, $\leq_{\text{1-log}}$.
- [I80] . . . stay complete **first-order** reductions.
- [V82] . . . stay complete via **projections**. (Non-uniform reductions where each bit of the output depends on at most one bit of the input).
- [I87] . . . stay complete via **first-order projections**, $\leq_{\mathrm{fop}}$.

Reductions

- [C71] SAT is NP complete via ptime Turing reductions.
- [K72] Many important problems are NP complete, via $\leq_{\mathrm{p}}$.
- [J73] …stay complete via **logspace** reductions, $\leq_{\mathrm{log}}$.
- [HIM78] …stay complete via **one-way logspace**, reductions, $\leq_{\text{1-log}}$.
- [I80] …stay complete **first-order** reductions.
- [V82] …stay complete via **projections**. (Non-uniform reductions where each bit of the output depends on at most one bit of the input).
- [I87] …stay complete via **first-order projections**, $\leq_{\mathrm{fop}}$.
- [L75] Artificial, non-complete problems can be constructed.

Reductions

- [C71] SAT is NP complete via ptime Turing reductions.
- [K72] Many important problems are NP complete, via $\leq_{\mathrm{p}}$.
- [J73] …stay complete via **logspace** reductions, $\leq_{\mathrm{log}}$.
- [HIM78] …stay complete via **one-way logspace**, reductions, $\leq_{\text{1-log}}$.
- [I80] …stay complete **first-order** reductions.
- [V82] …stay complete via **projections**. (Non-uniform reductions where each bit of the output depends on at most one bit of the input).
- [I87] …stay complete via **first-order projections**, $\leq_{\mathrm{fop}}$.
- [L75] Artificial, non-complete problems can be constructed.
- **Dichotomy:** "Natural" problems are complete for important compexity classes [FV99, S78, ABISV09].

Isomorphism Conjecture

- [BH77] **Isomorphism Conjecture:** “All NP complete sets via ptime many-one reductions, $\leq_{\mathrm{p}}$, are polynomial-time isomorphic.”

Isomorphism Conjecture

- [BH77] **Isomorphism Conjecture:** "All NP complete sets via ptime many-one reductions, $\leq_{\mathrm{p}}$, are polynomial-time isomorphic."
- [ABI93] **fop Isomorphism Thm.** All NP complete sets via $\leq_{\mathrm{fop}}$ are first-order isomorphic. Also true for L, NL, P, PSPACE, etc.

fop Isomorphism Theorem proof sketch

[BH77] **Observation:** All the NP complete sets in [GJ] are p-isomorphic.

fop Isomorphism Theorem proof sketch

[BH77] **Observation:** All the NP complete sets in [GJ] are p-isomorphic.

Schröder-Bernstein Thm. Let A and B be any two sets and suppose that $f : A \overset{1:1}{\to} B$ and $g : B \overset{1:1}{\to} A$. Then there exists $h : A \underset{\text{onto}}{\overset{1:1}{\to}} B$.

fop Isomorphism Theorem proof sketch

[BH77] **Observation:** All the NP complete sets in [GJ] are p-isomorphic.

Schröder-Bernstein Thm. Let A and B be any two sets and suppose that $f : A \overset{1:1}{\to} B$ and $g : B \overset{1:1}{\to} A$. Then there exists $h : A \underset{\text{onto}}{\overset{1:1}{\to}} B$. $\qquad (|A| \leq |B| \;\wedge\; |B| \leq |A| \;\to\; |A| = |B|)$

fop Isomorphism Theorem proof sketch

[BH77] **Observation:** All the NP complete sets in [GJ] are p-isomorphic.

Schröder-Bernstein Thm. Let A and B be any two sets and suppose that $f : A \overset{1:1}{\to} B$ and $g : B \overset{1:1}{\to} A$. Then there exists $h : A \underset{\text{onto}}{\overset{1:1}{\to}} B$. $\qquad (|A| \leq |B| \ \wedge \ |B| \leq |A| \ \rightarrow \ |A| = |B|)$

Proof: For $a, c \in A \cup B$, say that a is an **ancestor** of c if we can go from a to c by applying a finite, non-zero, number of applications of f and g.

fop Isomorphism Theorem proof sketch

[BH77] **Observation:** All the NP complete sets in [GJ] are p-isomorphic.

Schröder-Bernstein Thm. Let A and B be any two sets and suppose that $f : A \overset{1:1}{\to} B$ and $g : B \overset{1:1}{\to} A$. Then there exists $h : A \underset{\text{onto}}{\overset{1:1}{\to}} B$. $\quad (|A| \le |B| \;\wedge\; |B| \le |A| \;\to\; |A| = |B|)$

Proof: For $a, c \in A \cup B$, say that a is an **ancestor** of c if we can go from a to c by applying a finite, non-zero, number of applications of f and g.

$$h(a) \stackrel{\text{def}}{=} \begin{cases} g^{-1}(a) & \text{if } a \text{ has an odd number of ancestors} \\ f(a) & \text{if } a \text{ has an even or infinite number of ancestors} \end{cases}$$

fop Isomorphism Theorem proof sketch

[BH77] **Observation:** All the NP complete sets in [GJ] are p-isomorphic.

Schröder-Bernstein Thm. Let A and B be any two sets and suppose that $f : A \overset{1:1}{\to} B$ and $g : B \overset{1:1}{\to} A$. Then there exists $h : A \underset{\text{onto}}{\overset{1:1}{\to}} B$. $\quad (|A| \le |B| \wedge |B| \le |A| \to |A| = |B|)$

Proof: For $a, c \in A \cup B$, say that a is an **ancestor** of c if we can go from a to c by applying a finite, non-zero, number of applications of f and g.

$$h(a) \stackrel{\text{def}}{=} \begin{cases} g^{-1}(a) & \text{if } a \text{ has an odd number of ancestors} \\ f(a) & \text{if } a \text{ has an even or infinite number of ancestors} \end{cases}$$

Thus, $\quad h : A \underset{\text{onto}}{\overset{1:1}{\to}} B \qquad \square$

[BH77] **Observation:** All the NP complete sets in [GJ] are p-isomorphic.

[BH77] **Observation:** All the NP complete sets in [GJ] are p-isomorphic.

Lemma: Let $f : A \leq_{\mathrm{p}} B$ and $g : B \leq_{\mathrm{p}} A$ where f and g are 1:1 length-increasing functions. Assume also that f and g have left inverses in FP. Then A is p-isomorphic to B.

[BH77] **Observation:** All the NP complete sets in [GJ] are p-isomorphic.

Lemma: Let $f : A \leq_{\mathrm{p}} B$ and $g : B \leq_{\mathrm{p}} A$ where f and g are 1:1 length-increasing functions. Assume also that f and g have left inverses in FP. Then A is p-isomorphic to B.

Proof: Since f, g are length-increasing, the ancestor chains are linear in length. Thus, the isomorphism, h, can be defined as in the SB Thm, but now it can be computed in ptime. $\square$

Def. $A \subseteq \Sigma^*$ has **p-time padding functions** if $\exists e, d \in \mathrm{FP}$ s.t.

1. $\forall w, x \in \Sigma^* \quad w \in A \leftrightarrow e(w, x) \in A$
2. $\forall w, x \in \Sigma^* \quad d(e(w, x)) = x$
3. $\forall w, x \in \Sigma^* \quad |e(w, x)| \geq |w| + |x|$.

Def. $A \subseteq \Sigma^*$ has **p-time padding functions** if $\exists e, d \in \mathrm{FP}$ s.t.

1. $\forall w, x \in \Sigma^* \quad w \in A \leftrightarrow e(w, x) \in A$
2. $\forall w, x \in \Sigma^* \quad d(e(w, x)) = x$
3. $\forall w, x \in \Sigma^* \quad |e(w, x)| \geq |w| + |x|$.

Example: for SAT: $e(w, x) \stackrel{\text{def}}{=} (w) \wedge \underbrace{C_1 \wedge \cdots \wedge C_{|x|}}$, where

$C_i = (y \vee \overline{y})$ if $x_i = 1$, else $(\overline{y} \vee y)$.

Def. $A \subseteq \Sigma^*$ has **p-time padding functions** if $\exists e, d \in \mathrm{FP}$ s.t.

1. $\forall w, x \in \Sigma^* \quad w \in A \leftrightarrow e(w,x) \in A$
2. $\forall w, x \in \Sigma^* \quad d(e(w,x)) = x$
3. $\forall w, x \in \Sigma^* \quad |e(w,x)| \geq |w| + |x|$.

Example: for SAT: $\quad e(w,x) \stackrel{\text{def}}{=} (w) \wedge \underbrace{C_1 \wedge \cdots \wedge C_{|x|}}$, where

$C_i = (y \vee \overline{y})$ if $x_i = 1$, else $(\overline{y} \vee y)$.

Lemma: If $A, B \in \mathrm{NPC}$ and have p-time padding functions, then they are inter-reducible via p-time invertible 1:1 length-increasing reductions.

Def. $A \subseteq \Sigma^*$ has **p-time padding functions** if $\exists e, d \in \mathrm{FP}$ s.t.

1. $\forall w, x \in \Sigma^* \quad w \in A \leftrightarrow e(w,x) \in A$
2. $\forall w, x \in \Sigma^* \quad d(e(w,x)) = x$
3. $\forall w, x \in \Sigma^* \quad |e(w,x)| \geq |w| + |x|$.

Example: for SAT: $e(w,x) \stackrel{\text{def}}{=} (w) \wedge \underbrace{C_1 \wedge \cdots \wedge C_{|x|}}$, where $C_i = (y \vee \overline{y})$ if $x_i = 1$, else $(\overline{y} \vee y)$.

Lemma: If $A, B \in \mathrm{NPC}$ and have p-time padding functions, then they are inter-reducible via p-time invertible 1:1 length-increasing reductions.

Lemma: All the NP complete sets in [GJ] have p-time padding functions.

Def. $A \subseteq \Sigma^*$ has **p-time padding functions** if $\exists e, d \in \mathrm{FP}$ s.t.

1. $\forall w, x \in \Sigma^* \quad w \in A \ \leftrightarrow \ e(w,x) \in A$
2. $\forall w, x \in \Sigma^* \quad d(e(w,x)) = x$
3. $\forall w, x \in \Sigma^* \quad |e(w,x)| \geq |w| + |x|$.

Example: for SAT: $e(w,x) \stackrel{\text{def}}{=} (w) \wedge \underbrace{C_1 \wedge \cdots \wedge C_{|x|}}$, where $C_i = (y \vee \overline{y})$ if $x_i = 1$, else $(\overline{y} \vee y)$.

Lemma: If $A, B \in \mathrm{NPC}$ and have p-time padding functions, then they are inter-reducible via p-time invertible 1:1 length-increasing reductions.

Lemma: All the NP complete sets in [GJ] have p-time padding functions.

Thus, all the NP complete sets in [GJ] are p-isomorphic. $\square$

fop Isomorphism Theorem proof sketch

fop Isomorphism Thm. All NP complete sets via $\leq_{\text{fop}}$ are first-order isomorphic. Also true for NC^1, sAC^1, L, NL, P, PSPACE, etc.

fop Isomorphism Theorem proof sketch

fop Isomorphism Thm. All NP complete sets via $\leq_{\text{fop}}$ are first-order isomorphic. Also true for NC^1, sAC^1, L, NL, P, PSPACE, etc.

Key Lemma: Let f be a first-order projection (fop) that is 1:1 and of arity at least 2, i.e., it at least squares the size. Then the following two predicates are first-order expressible concerning a structure, $\mathcal{A}$:

1. $\mathrm{IE}(\mathcal{A})$, meaning that $f^{-1}(\mathcal{A})$ exists.
2. $\#\mathrm{Ancestors}(\mathcal{A}, r)$, meaning $\mathcal{A}$ has exactly r ancestors.

fop Isomorphism Theorem proof sketch

fop Isomorphism Thm. All NP complete sets via $\leq_{\text{fop}}$ are first-order isomorphic. Also true for NC^1, sAC^1, L, NL, P, PSPACE, etc.

Key Lemma: Let f be a first-order projection (fop) that is 1:1 and of arity at least 2, i.e., it at least squares the size. Then the following two predicates are first-order expressible concerning a structure, $\mathcal{A}$:

1. IE($\mathcal{A}$), meaning that $f^{-1}(\mathcal{A})$ exists.
2. #Ancestors($\mathcal{A}, r$), meaning $\mathcal{A}$ has exactly r ancestors.

The rest of the proof is similar to proof from [BH77]. $\square$

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

- **Morally**, the **BH Isomorphism Conjecture** is **true**.

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

- **Morally**, the **BH Isomorphism Conjecture** is **true**.
- Each **nice complexity class** has exactly **one complete problem**.

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

- **Morally**, the **BH Isomorphism Conjecture** is **true**.
- Each **nice complexity class** has exactly **one complete problem**.
- **Dichotomy Phenomenon: "Natural"** computational problems tend to be **complete via fops** for one of our **favorite complexity classes**.

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

- **Morally**, the **BH Isomorphism Conjecture** is **true**.
- Each **nice complexity class** has exactly **one complete problem**.
- **Dichotomy Phenomenon: "Natural"** computational problems tend to be **complete via fops** for one of our **favorite complexity classes**.
- Great for **Algorithms** and **Complexity Theory**!

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

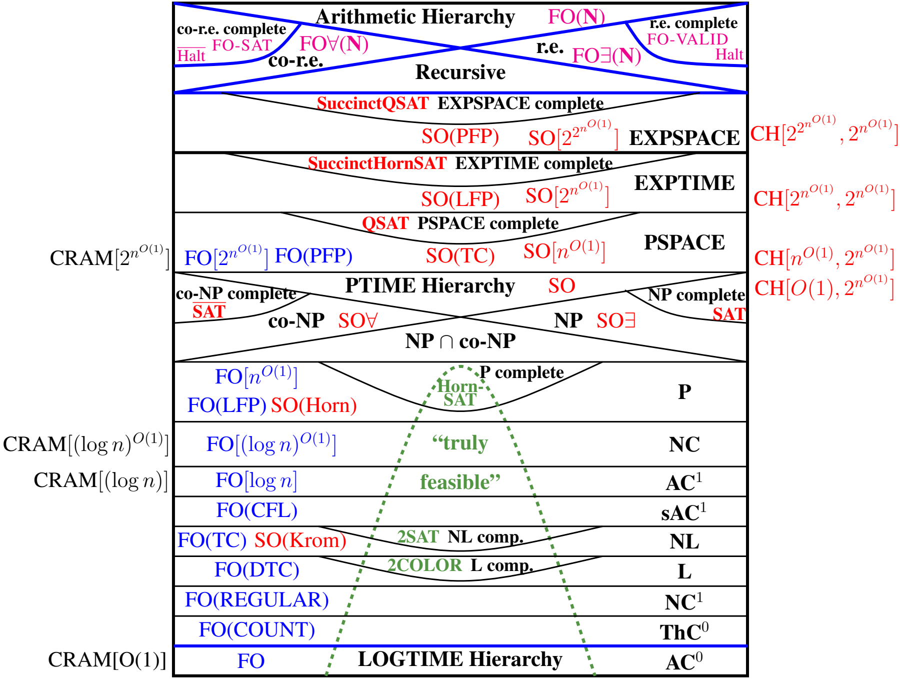
- **Morally**, the **BH Isomorphism Conjecture** is **true**.
- Each **nice complexity class** has exactly **one complete problem**.
- **Dichotomy Phenomenon: "Natural"** computational problems tend to be **complete via fops** for one of our **favorite complexity classes**.
- Great for **Algorithms** and **Complexity Theory**!
- But not true in general [L75].

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

- **Morally**, the **BH Isomorphism Conjecture** is **true**.
- Each **nice complexity class** has exactly **one complete problem**.
- **Dichotomy Phenomenon: "Natural"** computational problems tend to be **complete via fops** for one of our **favorite complexity classes**.
- Great for **Algorithms** and **Complexity Theory**!
- But not true in general [L75].
- Why does this seem to occur?

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

- **Morally**, the **BH Isomorphism Conjecture** is **true**.
- Each **nice complexity class** has exactly **one complete problem**.
- **Dichotomy Phenomenon: "Natural"** computational problems tend to be **complete via fops** for one of our **favorite complexity classes**.
- Great for **Algorithms** and **Complexity Theory**!
- But not true in general [L75].
- Why does this seem to occur?
- **Logical** and **Algebraic** reasons, e.g., CSP.


Arithmetic Hierarchy
FO(N)
co-r.e. complete
r.e. complete
FO-SAT
FO∀(N)
FO-VALID
Halt
co-r.e.
r.e.
FO∃(N)
Halt
Recursive
SuccinctQSAT EXPSPACE complete
SO(PFP)
SO[$2^{2^{n^{O(1)}}}$]
EXPSPACE
CH[$2^{2^{n^{O(1)}}}, 2^{n^{O(1)}}$]
SuccinctHornSAT EXPTIME complete
EXPTIME
SO(LFP)
SO[$2^{n^{O(1)}}$]
CH[$2^{n^{O(1)}}, 2^{n^{O(1)}}$]
QSAT PSPACE complete
PSPACE
CRAM[$2^{n^{O(1)}}$]
FO[$2^{n^{O(1)}}$] FO(PFP)
SO(TC)
SO[$n^{O(1)}$]
CH[$n^{O(1)}, 2^{n^{O(1)}}$]
PTIME Hierarchy
SO
CH[$O(1), 2^{n^{O(1)}}$]
co-NP complete
NP complete
SAT
SAT
co-NP SO∀
NP SO∃
NP ∩ co-NP
FO[$n^{O(1)}$]
P complete
Horn-SAT
P
FO(LFP) SO(Horn)
CRAM[$(\log n)^{O(1)}$]
FO[$(\log n)^{O(1)}$]
“truly
NC
CRAM[$(\log n)$]
FO[$\log n$]
feasible”
AC^1
FO(CFL)
sAC^1
FO(TC) SO(Krom)
2SAT NL comp.
NL
FO(DTC)
2COLOR L comp.
L
FO(REGULAR)
NC^1
FO(COUNT)
ThC^0
CRAM[O(1)]
FO
LOGTIME Hierarchy
AC^0

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

Some were unhappy with the fop Iso Thm because of a **mismatch**: fop more restrictive than fo.

fop Isomorphism Thm. For nice complexity classes, all complete sets via fops are first-order isomorphic.

Some were unhappy with the fop Iso Thm because of a **mismatch**: fop more restrictive than fo.

First-Order Isomorphism Theorem [Agrawal01] For nice complexity classes, all complete sets via fops are first-order isomorphic.