# Parallel Computation and Inductive Definitions

Neil Immerman

College of Computer and Information Sciences
University of Massachusetts, Amherst
Amherst, MA, USA

people.cs.umass.edu/~immerman

# Addition is First-Order

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

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

# Addition is First-Order

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

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

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

# Addition is First-Order

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

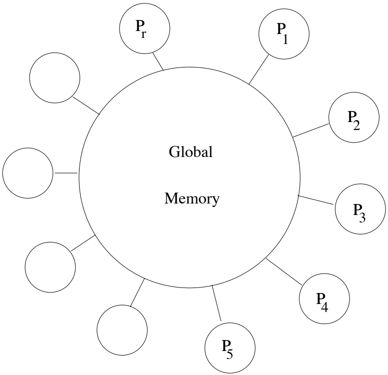
$$\begin{array}{lcccccc} A & & a_1 & a_2 & \ldots & a_{n-1} & a_n \\ B & + & b_1 & b_2 & \ldots & b_{n-1} & b_n \\ \hline S & & s_1 & s_2 & \ldots & s_{n-1} & s_n \end{array}$$

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

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

# Parallel Machines:

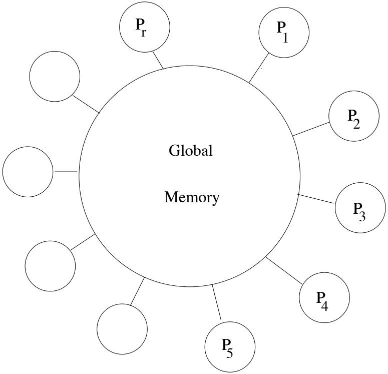
$\text{CRAM}[t(n)] \; = \text{CRCW-PRAM-TIME}[t(n)]\text{-HARD}[n^{O(1)}]$



# 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.

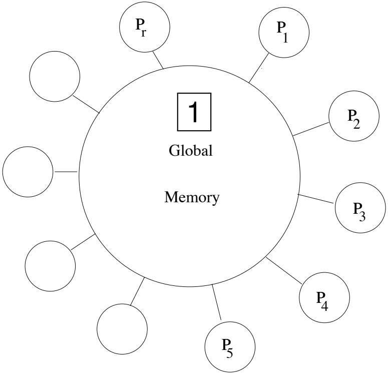


# Parallel Machines: Quantifiers are Parallel

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

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

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

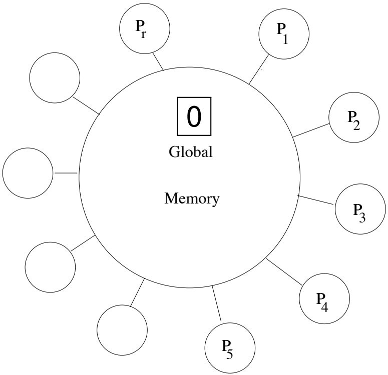


# Parallel Machines: Quantifiers are Parallel

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

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

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



FO

=

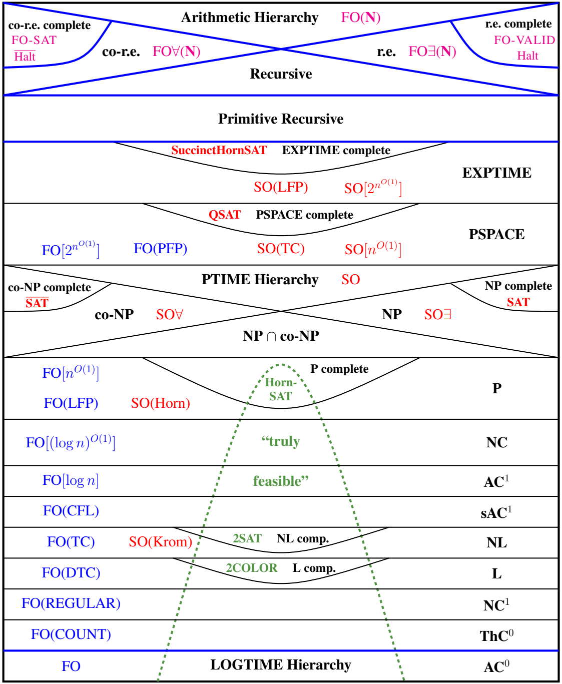
CRAM[1]

=

$AC^0$

=

Logarithmic-Time Hierarchy

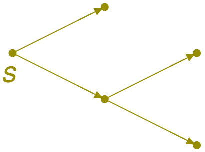


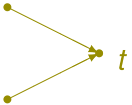
# Inductive Definitions and Least Fixed Point

$$\text{REACH} = \{G, s, t \mid s \xrightarrow{\star} t\}$$

# Inductive Definitions and Least Fixed Point

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



# Inductive Definitions and Least Fixed Point

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

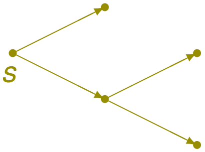
$$\text{REACH} = \left\{ G, s, t \;\middle|\; s \stackrel{\star}{\to} t \right\} \qquad \text{REACH} \notin \mathrm{FO}$$

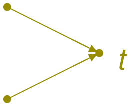
# Inductive Definitions and Least Fixed Point

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

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

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



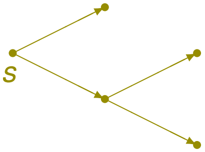
# Inductive Definitions and Least Fixed Point

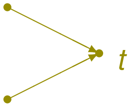
$$E^\star(x,y) \stackrel{\text{def}}{=} x = y \ \vee\ E(x,y) \ \vee\ \exists z(E^\star(x,z) \wedge E^\star(z,y))$$

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

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

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



# Inductive Definitions and Least Fixed Point

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

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

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

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

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

# Inductive Definitions and Least Fixed Point

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

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

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

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

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

# 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 $\mathrm{P}$.

# 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 $\mathrm{P}$.

**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 $\mathrm{P}$.

**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 $\mathrm{P}$.

**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 $\mathrm{P}$.

**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 $n = |V^G|$.

# 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 $\mathrm{P}$.

**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 $\varphi$.

# 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 $\mathrm{P}$.

**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 $\varphi$.

**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 $\mathrm{P}$.

**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 $\varphi$.

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

# 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 $\mathrm{P}$.

**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 $\varphi$.

**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 $\mathrm{P}$.

**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 $\varphi$.

**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 $\mathrm{P}$.

**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 $\varphi$.

**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 $\mathrm{P}$.

**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 $\varphi$.

**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

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

$$I^1 = \varphi_{tc}^G(\emptyset) = \{(a,b) \in V^G \times V^G \mid \mathrm{dist}(a,b) \leq 1\}$$

# Inductive Definition of Transitive Closure

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

# Inductive Definition of Transitive Closure

$$
\begin{aligned}
\varphi_{tc}(R, x, y) &\equiv x = y \lor E(x, y) \lor \exists z (R(x, z) \land R(z, y)) \\
I^1 = \varphi_{tc}^G(\emptyset) &= \{(a, b) \in V^G \times V^G \mid \mathrm{dist}(a, b) \le 1\} \\
I^2 = (\varphi_{tc}^G)^2(\emptyset) &= \{(a, b) \in V^G \times V^G \mid \mathrm{dist}(a, b) \le 2\} \\
I^3 = (\varphi_{tc}^G)^3(\emptyset) &= \{(a, b) \in V^G \times V^G \mid \mathrm{dist}(a, b) \le 4\}
\end{aligned}
$$

# Inductive Definition of Transitive Closure

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

# Inductive Definition of Transitive Closure

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

# Inductive Definition of Transitive Closure

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

# Inductive Definition of Transitive Closure

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

**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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

# parallel time = inductive depth = QB iteration

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

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

# 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 $\mathrm{QB}$ 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 $\mathrm{QB}$ 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 $\mathrm{QB}$ 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.

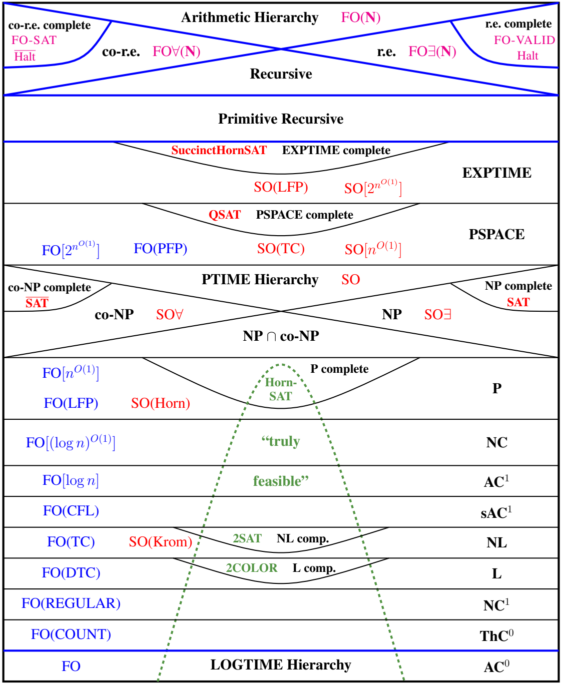
$\mathrm{CRAM}[t(n)] \subseteq \mathrm{FO}[t(n)]$: Inductively define new state of every bit of every register of every processor in terms of this global state at the previous time step. $\square$

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

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

For $t(n)$ poly bdd,

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



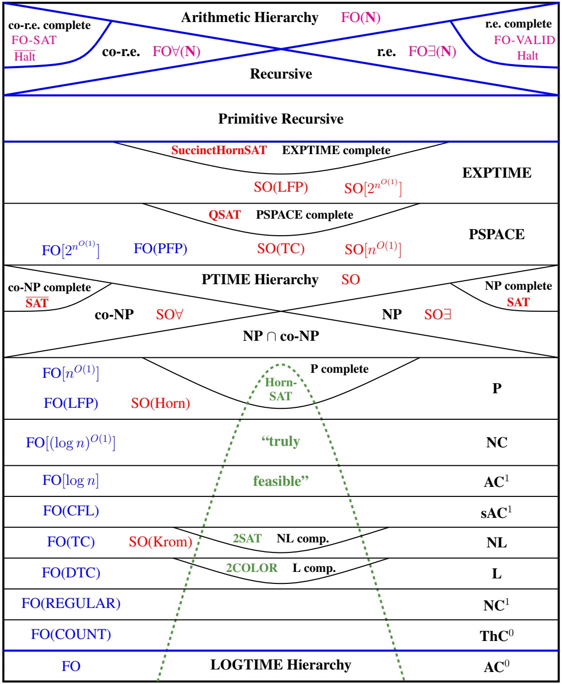
Remember that

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.

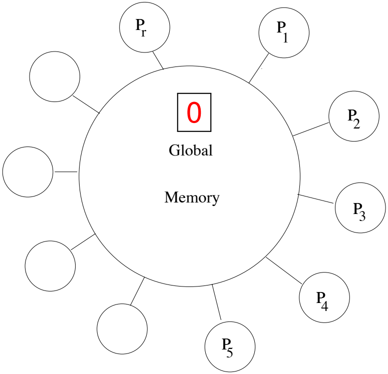
The proof is just a more detailed look at $\mathrm{CRAM}[t(n)] = \mathrm{FO}[t(n)]$.

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

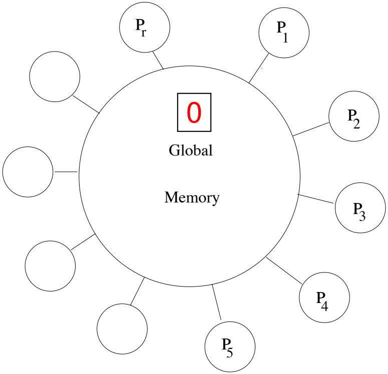
Given $\varphi$ with $n$ variables and $m$ clauses, is $\varphi \in$ 3-SAT?



# SO: Parallel Machines with Exponential Hardware

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

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

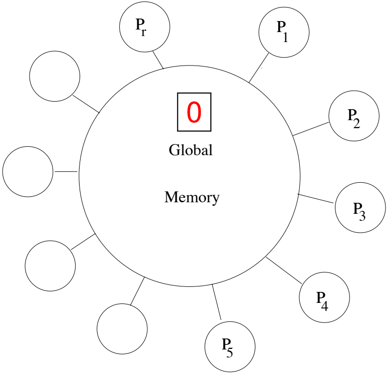


# SO: Parallel Machines with Exponential Hardware

Given $\varphi$ 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.

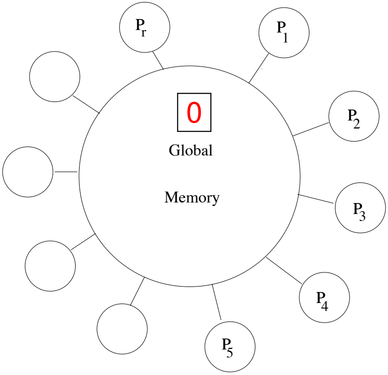


# SO: Parallel Machines with Exponential Hardware

Given $\varphi$ 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 $\varphi$ true, then $\varphi \in$ 3-SAT,

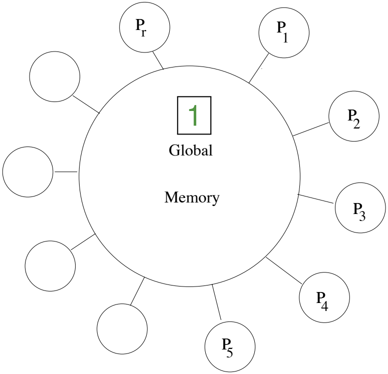


# SO: Parallel Machines with Exponential Hardware

Given $\varphi$ 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 $\varphi$ 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 $\mathrm{FO}[t(n)] = \mathrm{CRAM}[t(n)]$. $\square$

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

**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.**

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

# 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

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

# 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.

# Parallel Time versus Amount of Hardware

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

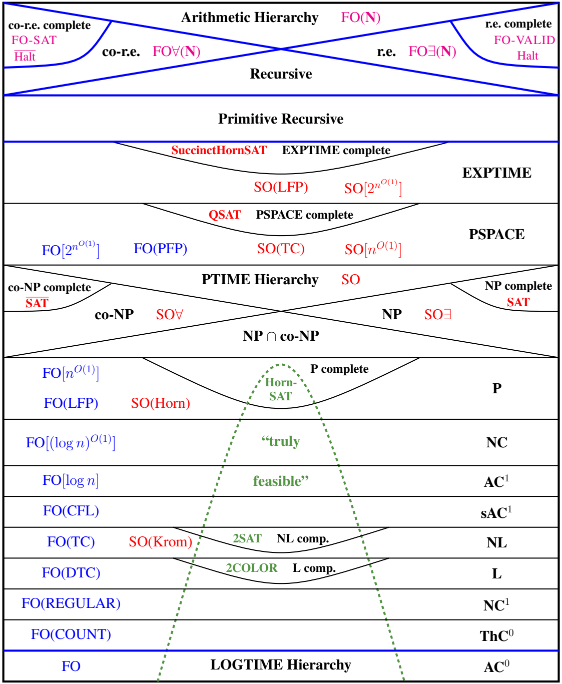
- We would love to understand this tradeoff.
- Is there such a thing as an inherently sequential problem?, i.e., is $\mathrm{NC} \neq \mathrm{P}$?

# Parallel Time versus Amount of Hardware

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

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

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



# 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 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.

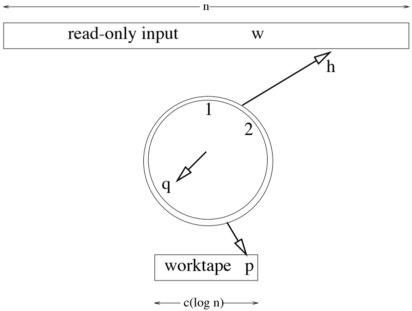
# Thm. REACH is complete for $\mathrm{NL} = \mathrm{NSPACE}[\log n]$.

**Proof:** Let $A \in \mathrm{NL}$, $A = \mathcal{L}(N)$, uses $c \log n$ bits of worktape.
Input $w$, $n = |w|$

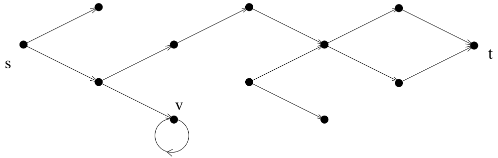
$$w \mapsto \mathrm{CompGraph}(N, w) \quad = \quad (V, E, s, t)$$

$$\begin{aligned}
V &= \left\{ \mathrm{ID} = \langle q, h, p \rangle \;\middle|\; q \in \mathrm{States}(N), h \leq n, |p| \leq c \lceil \log n \rceil \right\} \\
E &= \left\{ (\mathrm{ID}_1, \mathrm{ID}_2) \;\middle|\; \mathrm{ID}_1(w) \underset{N}{\longrightarrow} \mathrm{ID}_2(w) \right\} \\
s &= \text{initial ID} \\
t &= \text{accepting ID}
\end{aligned}$$

# NSPACE[log $n$] Turing Machine



**Claim.** $w \in \mathcal{L}(N) \iff \mathrm{CompGraph}(N, w) \in \mathsf{REACH}$




$\square$

**Cor:** $\mathrm{NL} \subseteq \mathrm{P}$

**Proof:** REACH $\in \mathrm{P}$

$\mathrm{P}$ is closed under (logspace) reductions.

i.e., $(B \in \mathrm{P} \quad \wedge \quad A \leq B) \quad \Rightarrow \quad A \in \mathrm{P}$ $\square$

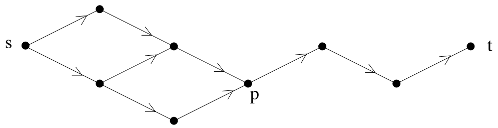
**Prop.**

$$\mathrm{NSPACE}[s(n)] \subseteq \mathrm{NTIME}[2^{O(s(n))}] \subseteq \mathrm{DSPACE}[2^{O(s(n))}]$$

**We can do much better!**

# Savitch's Theorem

$$\text{REACH} \in \text{DSPACE}(\log n)^2$$




**proof:**

$$\begin{array}{rcl} G \in \text{REACH} & \Leftrightarrow & G \models \text{PATH}_n(s,t) \\ \text{PATH}_1(x,y) & \equiv & x = y \ \vee \ E(x,y) \\ \text{PATH}_{2d}(x,y) & \equiv & \exists z \, (\text{PATH}_d(x,z) \ \wedge \ \text{PATH}_d(z,y)) \end{array}$$

$S_n(d)$ = space to check paths of dist. $d$ in $n$-nodegraphs

$$\begin{array}{rcl} S_n(n) & = & \log n + S_n(n/2) \\ & = & O((\log n)^2) \end{array}$$

$\square$

# Savitch's Theorem

$$\mathrm{DSPACE}[s(n)] \;\subseteq\; \mathrm{NSPACE}[n] \;\subseteq\; \mathrm{DSPACE}[(s(n))^2]$$

**proof:** Let $A \in \mathrm{NSPACE}[s(n)]$; $\quad A = \mathcal{L}(N)$

$$w \in A \qquad \Leftrightarrow \qquad \mathrm{CompGraph}(N, w) \in \text{REACH}$$

$$|w| = n; \qquad |\mathrm{CompGraph}(N, w)| = 2^{O(s(n))}$$

Testing if $\mathrm{CompGraph}(N, w) \in$ REACH takes space,

$$\begin{aligned} (\log(|\mathrm{CompGraph}(N, w)|))^2 &= (\log(2^{O(s(n))}))^2 \\ &= O((s(n))^2) \end{aligned}$$

From $w$ build $\mathrm{CompGraph}(N, w)$ in $\mathrm{DSPACE}$[s(n)]. $\square$

# $\overline{\text{REACH}} \in \mathrm{NL}$

**proof:** Fix $G$, let $N_d = \left|\{v \mid \text{distance}(s,v) \leq d\}\right|$

**Claim:** The following problems are in $\mathrm{NL}$:

1. $\mathrm{dist}(x,d)$: distance$(s,x) \leq d$
2. NDIST$(x,d;m)$: if $m = N_d$ then $\neg\mathrm{dist}(x,d)$

**proof:**

1. Guess the path of length $\leq d$ from $s$ to $x$.
2. Guess $m$ vertices, $v \neq x$, with $\mathrm{dist}(v,d)$.

$c := 0$;
**for** $v := 1$ to $n$ **do** $\{$ // nondeterministically
    ( $\mathrm{dist}(v,d)$ && $v \neq x$; $c{+}{+}$ ) $\quad ||$
    ( no-op )
$\}$
**if** $(c == m)$ **then** ACCEPT

$\square$

## **Claim.** We can compute $N_d$ in $\mathrm{NL}$.

**proof:** By induction on $d$.

**Base case:** $N_0 = 1$

**Inductive step:** Suppose we have $N_d$.

1. $c := 0;$
2. **for** $v := 1$ to $n$ **do** $\{$ // nondeterministically
3. $(\ \mathrm{dist}(v, d+1); c{+}{+})\quad ||$
4. $(\forall z\, (\mathrm{NDIST}(z, d; N_d)\ \vee\ (z \neq v \wedge \neg E(z, v))))$
5. $\}$
6. $N_{d+1} := c$

$\square$

$$G \in \overline{\mathrm{REACH}} \quad \Leftrightarrow \quad \mathrm{NDIST}(t, n; N_n) \qquad \square$$

**Thm.** $\mathrm{NSPACE}[s(n)] = \text{co-}\mathrm{NSPACE}[s(n)]$.

**proof:** Let $A \in \mathrm{NSPACE}[s(n)]$; $A = \mathcal{L}(N)$

$$w \in A \quad \Leftrightarrow \quad \mathrm{CompGraph}(N, w) \in \mathrm{REACH}$$

$$|w| = n; \qquad |\mathrm{CompGraph}(N, w)| = 2^{O(s(n))}$$

Testing if $\mathrm{CompGraph}(N, w) \in \overline{\mathrm{REACH}}$ takes space,

$$\begin{aligned} \log(|\mathrm{CompGraph}(N, w)|) &= \log(2^{O(s(n))}) \\ &= O(s(n)) \end{aligned} \qquad \square$$

# 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: $\mathrm{FO}$.

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

- [Sipser]: strict first-order alternation hierarchy: $\mathrm{FO}$.
- [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: $\mathrm{FO}$.
- [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: $\mathrm{FO}$.
- [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 $t(n)$, $\mathrm{FO}[t(n)] \;=\; \mathrm{CRAM}[t(n)]$

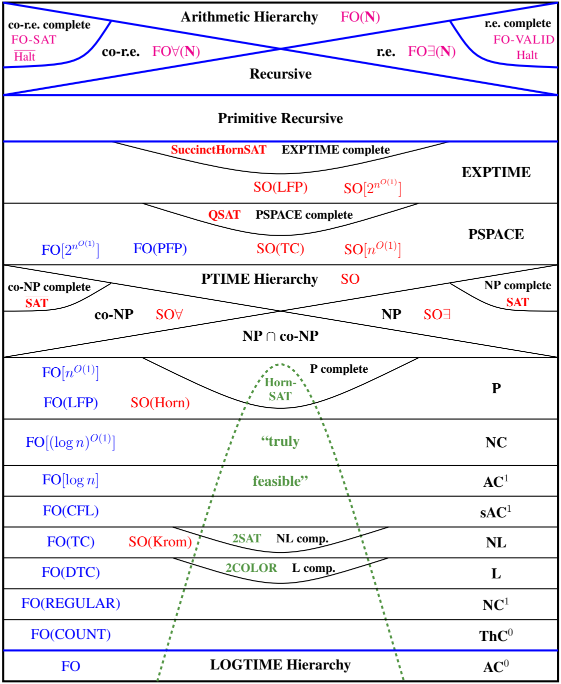
**Fact:** For $k = 1, 2, \ldots,$ $\mathrm{VAR}[k+1] \;=\; \mathrm{DSPACE}[n^k]$

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