Towards Capturing Order-Independent P

Neil Immerman

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

people.cs.umass.edu/~immerman
Descriptive Complexity

Query
\[ q_1 \quad q_2 \quad \cdots \quad q_n \]

\[ \mapsto \quad \text{Computation} \quad \mapsto \]

Answer
\[ a_1 \quad a_2 \quad \cdots \quad a_i \quad \cdots \quad a_m \]

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.

Neil Immerman
Towards Capturing Order-Independent P
Restrict attention to the complexity of computing individual bits of the output, i.e., decision problems.
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 $S$?
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 $S$?
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

\[ H = (\{a, b, c\}, \leq, E^H, R^H, G^H, B^H) \]

Colored

Graph

\[ H \]

\[
\begin{array}{c}
\text{a} \\
\text{c} \\
\text{b}
\end{array}
\]
Think of the Input as a Finite Logical Structure

\[ H = (\{a, b, c\}, \leq, E^H, R^H, G^H, B^H) \]

Colored Graph \[ E^H = \{(a, b), (b, a), (b, c), (c, b), (c, a), (a, c)\} \]
Think of the Input as a Finite Logical Structure

\[ H = (\{a, b, c\}, \leq, E^H, R^H, G^H, B^H) \]

Colored

Colored \[ E^H = \{(a, b), (b, a), (b, c), (c, b), (c, a), (a, c)\} \]

Graph

Graph \[ R^H = \{a\} \]

\[ G^H = \{b\} \]

\[ B^H = \{c\} \]
Think of the Input as a Finite Logical Structure

\[ H = (\{a, b, c\}, \leq, E^H, R^H, G^H, B^H) \]

\[ \leq^H = \{(a, a), (a, b), (a, c), (b, b), (b, c), (c, c)\} \]

Colored \[ E^H = \{(a, b), (b, a), (b, c), (c, b), (c, a), (a, c)\} \]

Graph \[ R^H = \{a\} \]

\[ G^H = \{b\} \]

\[ B^H = \{c\} \]
First-Order Logic

input symbols:  $E, R, Y, B, \ldots$
variables:  $x, y, z, \ldots$
boolean connectives:  $\land, \lor, \neg$
quantifiers:  $\forall, \exists$
numeric symbols:  $=, \leq, +, \times, \min, \max$

$\alpha \equiv \forall x \exists y \ E(x, y)$

$\beta \equiv \forall x y \ (\neg E(x, x) \land (E(x, y) \rightarrow E(y, x)))$

$\gamma \equiv \forall x ((\forall y \ x \leq y) \rightarrow R(x))$
First-Order Logic

input symbols:  $E, R, Y, B, \ldots$
variables:  $x, y, z, \ldots$
boolean connectives:  $\land, \lor, \neg$
quantifiers:  $\forall, \exists$
numeric symbols:  $=, \leq, +, \times, \min, \max$

$\alpha \equiv \forall x \exists y \ E(x, y)$
$\beta \equiv \forall xy \ (\neg E(x, x) \land (E(x, y) \rightarrow E(y, x)))$
$\gamma \equiv \forall x ((\forall y \ x \leq y) \rightarrow R(x))$

In this setting, with the structure of interest being the finite input, FO is a weak complexity class.
First-Order Logic

input symbols: $E, R, Y, B, \ldots$

variables: $x, y, z, \ldots$

boolean connectives: $\land, \lor, \neg$

quantifiers: $\forall, \exists$

numeric symbols: $=, \leq, +, \times, \text{min}, \text{max}$

$\alpha \equiv \forall x \exists y \ E(x, y)$

$\beta \equiv \forall xy \ (\neg E(x, x) \land (E(x, y) \rightarrow E(y, x)))$

$\gamma \equiv \forall x ((\forall y \ x \leq y) \rightarrow R(x))$

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

It is easy to test if input, $H$, satisfies $\alpha$ ($H \models \alpha$).
First-Order Logic

\[ H \quad a \leq b \leq c \]

\[ G \quad 1 \leq 2 \leq 3 \]

\[ \alpha \equiv \forall x \exists y \ E(x, y) \]

\[ \beta \equiv \forall xy (\neg E(x, x) \land (E(x, y) \rightarrow E(y, x))) \]

\[ \gamma \equiv \forall x ((\forall y \ x \leq y) \rightarrow R(x)) \]
First-Order Logic

\[ H \quad a \leq b \leq c \]

\[ G \quad 1 \leq 2 \leq 3 \]

\[ H \models \alpha \land \beta \land \gamma \]

\[ \alpha \equiv \forall x \exists y \ E(x, y) \]

\[ \beta \equiv \forall xy \ (\neg E(x, x) \land (E(x, y) \rightarrow E(y, x))) \]

\[ \gamma \equiv \forall x ((\forall y x \leq y) \rightarrow R(x)) \]
First-Order Logic

\[ H \quad a \leq b \leq c \]

\[ G \quad 1 \leq 2 \leq 3 \]

\[ H \models \alpha \land \beta \land \gamma \]

\[ G \models \alpha \land \beta \land \neg \gamma \]

\[ \alpha \equiv \forall x \exists y \ E(x, y) \]

\[ \beta \equiv \forall xy \ (\neg E(x, x) \land (E(x, y) \rightarrow E(y, x))) \]

\[ \gamma \equiv \forall x ((\forall y x \leq y) \rightarrow R(x)) \]
First-Order Logic

\[ H \quad a \leq b \leq c \]

\[ G \quad 1 \leq 2 \leq 3 \]

\[ H \models \alpha \land \beta \land \gamma \]

\[ G \models \alpha \land \beta \land \neg \gamma \]

\[ \alpha \equiv \forall x \exists y \ E(x, y) \]

\[ \beta \equiv \forall xy (\neg E(x, x) \land (E(x, y) \rightarrow E(y, x))) \]

\[ \gamma \equiv \forall x ((\forall y x \leq y) \rightarrow R(x)) \]

\[ \alpha \text{ and } \beta \text{ are order independent; } \quad \gamma \text{ is order dependent} \]
\[ \Phi_{3\text{color}} \equiv \exists R^1 \ G^1 \ B^1 \ \forall x \ y \ ((R(x) \lor G(x) \lor B(x)) \land (E(x, y) \rightarrow \neg(R(x) \land R(y)) \land \neg(G(x) \land G(y)) \land \neg(B(x) \land B(y)))) \]
Fagin’s Theorem: \( \text{NP} = \text{SO}\exists \)

\[
\Phi_{3\text{color}} \equiv \exists R^1 \ G^1 \ B^1 \ \forall x \ y \ ((R(x) \lor G(x) \lor B(x)) \ \land \ (E(x, y) \rightarrow
\neg (R(x) \land R(y)) \land \neg (G(x) \land G(y)) \land \neg (B(x) \land B(y))))
\]
Towards Capturing Order-Independent P
Inductive Definitions and Least Fixed Point

\[ \text{REACH} = \{ G, s, t \mid s \xrightarrow{*} t \} \]
Inductive Definitions and Least Fixed Point

\[ \text{REACH} = \{ G, s, t \mid s \xrightarrow{*} t \} \]

\[ \text{REACH} \notin \text{FO} \]
Inductive Definitions and Least Fixed Point

\[ E^*(x, y) \overset{\text{def}}{=} x = y \lor E(x, y) \lor \exists z(E^*(x, z) \land E^*(z, y)) \]

\[
\text{REACH} = \{ G, s, t \mid s \xrightarrow{*} t \}
\]

\[
\text{REACH} \not\in \text{FO}
\]
Inductive Definitions and Least Fixed Point

\[ E^*(x, y) \overset{\text{def}}{=} x = y \lor E(x, y) \lor \exists z(E^*(x, z) \land E^*(z, y)) \]

\[ \varphi_{\text{tc}}(R, x, y) \overset{\text{def}}{=} x = y \lor E(x, y) \lor \exists z(R(x, z) \land R(z, y)) \]

\[ \text{REACH} = \{ G, s, t \mid s \xrightarrow{\ast} t \} \]

\[ \text{REACH} \not\in \text{FO} \]
Inductive Definitions and Least Fixed Point

\[ E^*(x, y) \overset{\text{def}}{=} x = y \lor E(x, y) \lor \exists z(E^*(x, z) \land E^*(z, y)) \]

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

\[ \varphi^G_{tc} : \text{binRel}(G) \to \text{binRel}(G) \]

\[ \text{monotone} \quad R \subseteq S \implies \varphi^G_{tc}(R) \subseteq \varphi^G_{tc}(S) \]

\[ \text{REACH} = \{ G, s, t \mid s \xrightarrow{*} t \} \]

\[ \text{REACH} \not\in \text{FO} \]
**Inductive Definitions and Least Fixed Point**

\[ E^*(x, y) \overset{\text{def}}{=} x = y \lor E(x, y) \lor \exists z(E^*(x, z) \land E^*(z, y)) \]

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

\[ \varphi^G_{tc} : \text{binRel}(G) \rightarrow \text{binRel}(G) \]

**monotone**

\[ R \subseteq S \Rightarrow \varphi^G_{tc}(R) \subseteq \varphi^G_{tc}(S) \]

\[ E^* = (\text{LFP}\varphi_{tc}) \]

\[ \text{REACH} = \{ G, s, t \mid s \xrightarrow{k} t \} \]

\[ \text{REACH} \notin \text{FO} \]
Inductive Definitions and Least Fixed Point

\[ E^*(x, y) \overset{\text{def}}{=} x = y \lor E(x, y) \lor \exists z(E^*(x, z) \land E^*(z, y)) \]

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

\[ \varphi^G_{tc} : \text{binRel}(G) \rightarrow \text{binRel}(G) \]

monotone  \hspace{1cm} R \subseteq S \implies \varphi^G_{tc}(R) \subseteq \varphi^G_{tc}(S) \]

\[ G \in \text{REACH} \iff G \models (\text{LFP}\varphi_{tc})(s, t) \]

\[ E^* = (\text{LFP}\varphi_{tc}) \]

\[ \text{REACH} = \{ G, s, t \mid s^* \rightarrow t \} \]

\[ \text{REACH} \not\in \text{FO} \]
LFP is a Polynomial Iteration Operator

**Thm.** \( P = \text{FO}(\text{LFP}) = \text{FO}[n^{O(1)}] \)

\( \text{FO}[n^{O(1)}] \) means for graphs with \( n \) vertices, the formula \( \varphi_n \) expressing the property has \( n^{O(1)} \) quantifiers, but only a **fixed number** of requantified **variables**, \( x_1, \ldots, x_k \), i.e., \( \varphi_n \in \mathcal{L}^k \).
**Thm.**  \( P = \text{FO}(\text{LFP}) = \text{FO}[n^{O(1)}] \)

Graphs are completely general structures, i.e., any structure can be encoded as a graph.

\( \text{FO}[n^{O(1)}] \) means for graphs with \( n \) vertices, the formula \( \varphi_n \) expressing the property has \( n^{O(1)} \) quantifiers, but only a **fixed number** of requantified **variables**, \( x_1, \ldots, x_k \), i.e, \( \varphi_n \in \mathcal{L}^k \).
**Thm.** \( P = \text{FO}(\text{LFP}) = \text{FO}[n^{O(1)}] \)

Graphs are completely general structures, i.e., any structure can be encoded as a graph. **Restrict to graphs.**

\( \text{FO}[n^{O(1)}] \) means for graphs with \( n \) vertices, the formula \( \varphi_n \) expressing the property has \( n^{O(1)} \) quantifiers, but only a **fixed number** of requantified **variables**, \( x_1, \ldots, x_k \), i.e, \( \varphi_n \in \mathcal{L}^k \).
LFP is a Polynomial Iteration Operator

**Thm.** \[ P = \text{FO}(\text{LFP}) = \text{FO}[n^{O(1)}] \]

Graphs are completely general structures, i.e., any structure can be encoded as a graph. **Restrict to graphs.**

\[ \text{FO}[n^{O(1)}] \] means for graphs with \( n \) vertices, the formula \( \varphi_n \) expressing the property has \( n^{O(1)} \) quantifiers, but only a **fixed number** of requantified variables, \( x_1, \ldots, x_k \), i.e, \( \varphi_n \in \mathcal{L}^k \).

Above **Thm** requires ordering relation, \( \leq \).
LFP is a Polynomial Iteration Operator

**Thm.** \( \mathbf{P} = \text{FO}(\text{LFP}) = \text{FO}[n^{O(1)}] \)

Graphs are completely general structures, i.e., any structure can be encoded as a graph. **Restrict to graphs.**

\( \text{FO}[n^{O(1)}] \) means for graphs with \( n \) vertices, the formula \( \varphi_n \) expressing the property has \( n^{O(1)} \) quantifiers, but only a **fixed number** of requantified **variables**, \( x_1, \ldots, x_k \), i.e, \( \varphi_n \in \mathcal{L}^k \).

Above **Thm** requires ordering relation, \( \leq \).

Necessary for encoding computation – inputs to computers are ordered.
**Thm.** \[ P = \text{FO}(\text{LFP}) = \text{FO}[n^{O(1)}] \]

Graphs are completely general structures, i.e., any structure can be encoded as a graph. **Restrict to graphs.**

\[ \text{FO}[n^{O(1)}] \] means for graphs with \( n \) vertices, the formula \( \varphi_n \) expressing the property has \( n^{O(1)} \) quantifiers, but only a **fixed number** of requantified variables, \( x_1, \ldots, x_k \), i.e, \( \varphi_n \in \mathcal{L}^k \).

Above **Thm** requires ordering relation, \( \leq \).

Necessary for encoding computation – inputs to computers are ordered.

Unnatural for graphs – the ordering of the vertices is irrelevant.
LFP is a Polynomial Iteration Operator

**Thm.** \( P = \text{FO}(\text{LFP}) = \text{FO}[n^{O(1)}] \)

Graphs are completely general structures, i.e., any structure can be encoded as a graph. **Restrict to graphs.**

\( \text{FO}[n^{O(1)}] \) means for graphs with \( n \) vertices, the formula \( \varphi_n \) expressing the property has \( n^{O(1)} \) quantifiers, but only a **fixed number** of requantified variables, \( x_1, \ldots, x_k \), i.e, \( \varphi_n \in \mathcal{L}^k \).

Above **Thm** requires ordering relation, \( \leq \).

Necessary for encoding computation – inputs to computers are ordered.

Unnatural for graphs – the ordering of the vertices is irrelevant.

**Wanted:** a language capturing Order-Independent P (OIP).
Want to Capture Order-Indepdendent P (OIP)

\[ \text{FO}(\text{LFP}) = \text{P} \]

\[ \text{FO}(\text{wo} \leq)\text{(LFP)} \subseteq \text{OIP} \]
Want to Capture Order-Independent P (OIP)

\[ FO(LFP) = P \]

\[ FO(\text{wo}\leq)(LFP) \subseteq OIP \]

\[ \text{EVEN} \overset{\text{def}}{=} \{ G \mid |V^G| \equiv 0 \pmod{2} \} \]
Want to Capture Order-Independent P (OIP)

\[ \text{FO}(\text{LFP}) = \text{P} \]

\[ \text{FO}(\text{wo} \leq)(\text{LFP}) \subseteq \text{OIP} \]

\[ \text{EVEN} \overset{\text{def}}{=} \{ G \mid |V^G| \equiv 0 \pmod{2} \} \]

\[ \text{EVEN} \in \text{OIP} - \text{FO}(\text{wo} \leq)(\text{LFP}). \]
Want to Capture Order-Independent P (OIP)

\[ \text{FO}(\text{LFP}) = \text{P} \]

\[ \text{FO}(\text{wo}\leq)(\text{LFP}) \subseteq \text{OIP} \]

\[ \text{EVEN} \overset{\text{def}}{=} \{ G \mid |V^G| \equiv 0 \pmod{2} \} \]

\[ \text{EVEN} \in \text{OIP} - \text{FO}(\text{wo}\leq)(\text{LFP}). \]

Thus, \[ \text{FO}(\text{wo}\leq)(\text{LFP}) \nsubseteq \text{OIP} \]
Want to Capture Order-Indepdendentent P (OIP)

\[ \text{FO}(\text{LFP}) = \text{P} \]

\[ \text{FO}(\text{wo} \leq)(\text{LFP}) \subseteq \text{OIP} \]

\[ \text{EVEN} \overset{\text{def}}{=} \{ G \mid |V^G| \equiv 0 \pmod{2} \} \]

\[ \text{EVEN} \in \text{OIP} - \text{FO}(\text{wo} \leq)(\text{LFP}). \]

Thus, \[ \text{FO}(\text{wo} \leq)(\text{LFP}) \nsubseteq \text{OIP} \]

How do we prove \( \text{EVEN} \not\in \text{FO}(\text{wo} \leq)(\text{LFP}) \)?
Ehrenfeucht-Fraïssé Game

$g^k_m(G, H)$  $m$ moves,  $k$ pebbles,  2 players

Samson: show a difference.
Delilah: preserve isomorphism.

For all $m$, $D$ wins $g^2_m(G, H)$; but $S$ wins $g^3_3(G, H)$. 
Ehrenfeucht-Fraïssé Game

$g^k_m(G, H)$

$m$ moves, $k$ pebbles, 2 players

Samson: show a difference.
Ehrenfeucht-Fraïssé Game

\( g^k_m(G, H) \) \( m \) moves, \( k \) pebbles, 2 players

**Samson**: show a difference. **Delilah**: preserve isomorphism.
Ehrenfeucht-Fraïssé Game

\( g^k_m(G, H) \) \( m \) moves, \( k \) pebbles, 2 players

**Samson**: show a difference.  **Delilah**: preserve isomorphism.
Ehrenfeucht-Fraïssé Game

\[ g^k_m(G, H) \] \( m \) moves, \( k \) pebbles, 2 players

**Samson**: show a difference.  **Delilah**: preserve isomorphism.
Ehrenfeucht-Fraïssé Game

$G^k_m(G,H)$  $m$ moves,  $k$ pebbles,  2 players

**Samson**: show a difference.  **Delilah**: preserve isomorphism.
Ehrenfeucht-Fraïssé Game

$g_m^k(G, H)$ $m$ moves, $k$ pebbles, 2 players

**Samson:** show a difference. **Delilah:** preserve isomorphism.
Ehrenfeucht-Fraïssé Game

$G^k_m(G, H)$ $m$ moves, $k$ pebbles, 2 players

**Samson**: show a difference.  **Delilah**: preserve isomorphism.
Ehrenfeucht-Fraïssé Game

$g^k_m(G, H)$  \( m \) moves,  \( k \) pebbles,  2 players

**Samson**: show a difference.  **Delilah**: preserve isomorphism.
Ehrenfeucht-Fraïssé Game

$g_m^k(G, H)$  $m$ moves,  $k$ pebbles,  2 players

**Samson**: show a difference.  **Delilah**: preserve isomorphism.

For all $m$, $D$ wins $g_m^2(G, H)$;

---

Neil Immerman  Towards Capturing Order-Independent P
Ehrenfeucht-Fraïssé Game

$\mathcal{G}^k_m(G, H)$  \( m \) moves,  \( k \) pebbles,  2 players

**Samson**: show a difference.  **Delilah**: preserve isomorphism.

For all \( m \), \( D \) wins \( \mathcal{G}^2_m(G, H) \);

\[\begin{array}{ccc}
G & & H \\
\text{---} & \text{---} & \text{---} \\
\text{---} & \text{---} & \text{---} \\
\text{---} & \text{---} & \text{---} \\
\text{---} & \text{---} & \text{---} \\
\end{array}\]
Ehrenfeucht-Fraïssé Game

$\mathcal{G}_m^k(G, H)$  
$m$ moves,  
$k$ pebbles,  
2 players

**Samson**: show a difference.  
**Delilah**: preserve isomorphism.  

For all $m$, $D$ wins $\mathcal{G}_m^2(G, H)$;
Ehrenfeucht-Fraïssé Game

$G^k_m(G, H)$ $m$ moves, $k$ pebbles, 2 players

**Samson**: show a difference. **Delilah**: preserve isomorphism.

For all $m$, D wins $G^2_m(G, H)$;
Ehrenfeucht-Fraïssé Game

$\mathcal{G}_m^k(G, H)$  $m$ moves,  $k$ pebbles,  2 players

**Samson**: show a difference.  **Delilah**: preserve isomorphism.
For all $m$, $D$ wins $\mathcal{G}_m^2(G, H)$; but $S$ wins $\mathcal{G}_3^3(G, H)$. 

---

![Diagram of graphs G and H with pebbles and moves marked.]
Notation: \( G \sim^k_m H \) means that Delilah has a winning strategy for \( G^k_m(G, H) \).
**Notation:** \( G \sim^k_m H \) means that Delilah has a winning strategy for \( G^k_m(G, H) \).

**Thm.** D has a winning strategy on the \( m \)-move, \( k \)-pebble game on \( G, H \) iff \( G \) and \( H \) agree on all formulas using \( k \) variables and quantifier depth \( m \).

\[
G \sim^k_m H \iff G \equiv^k_m H
\]
Thm. \textbf{EVEN} requires $n + 1$ variables without ordering. Thus \textbf{EVEN} \not\in \text{FO}(\text{wo}\leq)(\text{LFP}).
Thm. \textbf{EVEN} requires $n + 1$ variables without ordering. Thus \textbf{EVEN} $\not\in \text{FO(wo}\leq\text{)}(\text{LFP})$.

proof:

\[ \begin{array}{c}
G_{2m} \\
\vdots \\
g_{2m} \\
h_{2m} \\
h_{2m+1}
\end{array} \quad \begin{array}{c}
\vdots \\
h_2 \\
h_1 \\
h_{2m} \\
h_{2m+1}
\end{array} \quad H_{2m+1} \]
Thm. \textbf{EVEN} requires $n + 1$ variables without ordering. Thus \textbf{EVEN} $\not\in \text{FO}(\text{wo}\leq)(\text{LFP})$.

proof:

\[ g_1 \quad x_1 \quad g_2 \quad \ldots \quad g_{2m} \quad H_{2m+1} \]

\[ h_1 \quad \ldots \quad h_{2m} \quad h_{2m+1} \]
Thm. **EVEN** requires $n + 1$ variables without ordering. Thus **EVEN** $\not\in \text{FO}(\text{wo} \leq)\text{(LFP)}$.

**proof:**

\[
\begin{array}{c}
g_1 \quad x_1 \\
g_2 \\
G_{2m} \\
g_{2m} \\
\hline
h_1 \quad x_1 \\
h_2 \\
H_{2m+1} \\
h_{2m} \\
h_{2m+1}
\end{array}
\]
Thm. **EVEN** requires \( n + 1 \) variables without ordering. Thus \( \text{EVEN} \not\in \text{FO(wo}\leq\text{)}(\text{LFP}) \).

**proof:**

\[
\begin{align*}
G_{2m} & : \\
g_1 & \sim x_1 \\
g_2 & \\
g_{2m} & \\
\vdots \\
H_{2m+1} & : \\
h_1 & \sim x_1 \\
h_2 & \sim x_2 \\
h_{2m} & \\
h_{2m+1} & 
\end{align*}
\]
**Thm.** \( \text{EVEN} \) requires \( n + 1 \) variables without ordering. Thus \( \text{EVEN} \not\in \text{FO(wo\leq)(LFP)} \).

**proof:**

\[ G_{2m} \quad : \quad h_{2m} \]

\[ g_1 \quad x_1 \quad \quad \quad \quad h_1 \quad x_1 \]

\[ g_2 \quad x_2 \quad \quad \quad \quad h_2 \quad x_2 \]

\[ g_{2m} \quad \quad \quad \quad h_{2m} \]

\[ \vdots \quad \quad \quad \quad \vdots \quad \quad \quad \quad \vdots \quad \quad \quad \quad \vdots \quad \quad \quad \quad \vdots \]

\[ \vdots \quad \quad \quad \quad \vdots \quad \quad \quad \quad \vdots \quad \quad \quad \quad \vdots \]

\[ H_{2m+1} \]

\[ h_{2m+1} \]
Thm. \textbf{EVEN} requires $n + 1$ variables without ordering. Thus \textbf{EVEN} $\not\in \text{FO}(\text{wo}\leq)(\text{LFP})$.

\textbf{proof:}

\[ G_{2m} : \]
\[ g_1 \quad x_1 \]
\[ g_2 \quad x_2 \]
\[ g_{2m} \quad x_{2m} \]

\[ H_{2m+1} : \]
\[ h_1 \quad x_1 \]
\[ h_2 \quad x_2 \]
\[ h_{2m} \]
\[ h_{2m+1} \]
Thm. **EVEN** requires $n + 1$ variables without ordering. Thus **EVEN** $\not\in \text{FO(wo}\leq\text{(LFP)).}$

**proof:**

$$G_{2m} \sim H_{2m+1}$$
Thm. **EVEN** requires \( n + 1 \) variables without ordering. Thus **EVEN** \( \not\in FO(\text{wo}\leq)(LFP) \).

**proof:**

\[
\begin{align*}
G_{2m} & \sim^{2m} H_{2m+1} \\
G_1 & \sim x_1 \\
G_2 & \sim x_2 \\
G_{2m} & \sim x_{2m} \\
H_1 & \sim x_1 \\
H_2 & \sim x_2 \\
H_{2m} & \sim x_{2m} \\
H_{2m+1} & \sim 
\end{align*}
\]
Add Counting to FO Logic

Two sorts: **Numbers**: \{0, 1, \ldots, n\}, \leq, Plus, Times and

**Vertices**: \{v_1, \ldots, v_n\}, E, C_1, C_2 \ldots
Two sorts: **Numbers**: \( \{0, 1, \ldots, n\} \), \( \leq \), Plus, Times and **Vertices**: \( \{v_1, \ldots, v_n\} \), \( E, C_1, C_2 \ldots \)

Combine with counting terms: \( \#x(\varphi(x)) \).
Add Counting to FO Logic

Two sorts: **Numbers**: \( \{0, 1, \ldots, n\} \), \( \leq \), Plus, Times and **Vertices**: \( \{v_1, \ldots, v_n\} \), \( E, C_1, C_2 \ldots \)

Combine with counting terms: \( \#x(\varphi(x)) \).

\[
\text{EVEN} \equiv \exists i \left( \text{Plus}(i, i, \#x(x = x)) \right)
\]
Two sorts: **Numbers**: \{0, 1, \ldots, n\}, \leq, Plus, Times and **Vertices**: \{v_1, \ldots, v_n\}, E, C_1, C_2 \ldots

Combine with counting terms: \#x(\varphi(x))

EVEN \equiv \exists i (\text{Plus}(i, i, \#x(x = x)))

Let \( C^k \) \( \overset{\text{def}}{=} \) FO\(^k\)(COUNT); \( \text{FPC} \) \( \overset{\text{def}}{=} \) FO(LFP, COUNT).
Add Counting to FO Logic

Two sorts: **Numbers**: \( \{0, 1, \ldots, n\}, \leq, \) Plus, Times and **Vertices**: \( \{v_1, \ldots, v_n\}, E, C_1, C_2 \ldots \)

Combine with counting terms: \( \#x(\varphi(x)). \)

\[
\text{EVEN} \equiv \exists i (\text{Plus}(i, i, \#x(x = x)))
\]

Let \( C^k \overset{\text{def}}{=} \text{FO}^k(\text{COUNT}); \quad \text{FPC} \overset{\text{def}}{=} \text{FO}(\text{LFP, COUNT}). \)

\[
\text{FO(wo}\leq)(\text{LFP}) \subseteq \not= \text{FPC} \subseteq \text{OIP}
\]
Stable Coloring of Vertices

Start with a colored graph, and repeatedly color each vertex by how many neighbors it has of each color.

Thm. Stable Coloring of Vertices = $C^2$ type.

Round $m$ of stable coloring is quantifier depth of $C^2$ formula.
Stable Coloring of Vertices

Start with a colored graph, and repeatedly color each vertex by how many neighbors it has of each color.
Stable Coloring of Vertices

Start with a colored graph, and repeatedly color each vertex by how many neighbors it has of each color.
Stable Coloring of Vertices

Start with a colored graph, and repeatedly color each vertex by how many neighbors it has of each color.

Thm. Stable Coloring of Vertices $= C_2^{\text{type.}}$ Round $m$ of stable coloring is quantifier depth of $C_2$ formula.
Stable Coloring of Vertices

Start with a colored graph, and repeatedly color each vertex by how many neighbors it has of each color.
Stable Coloring of Vertices

Start with a colored graph, and repeatedly color each vertex by how many neighbors it has of each color.

Thm. Stable Coloring of Vertices $= C^2$ type.

Round $m$ of stable coloring is quantifier depth of $C^2$ formula.
The Good News: Upper Bounds

**Thm.** [Babai, Erdos, Selkow] With high probability, after four iterations of stable coloring, each vertex of a random graph has a unique color, i.e., the $C_4^2$-type of each vertex is unique.

Thus, for almost all graphs, there is a linear time algorithm to canonize the graph, i.e., sort the vertices by their $C_4^2$-type, so that two graphs are isomorphic iff their canonical forms are equal.

With high probability, $G \sim H$ iff $G \equiv C_4^2 H$.

Thus, Graph Isomorphism (GI) is linear time for random graphs.

In general the complexity of GI is unknown.

**Thm.** [Babai, 2015] GI $\in \text{DTIME} \left[ n \log^7 n \right]$. (Before this it was only known that GI $\in \text{DTIME} \left[ n \sqrt[2]{n} \right]$.)
The Good News: Upper Bounds

**Thm.** [Babai, Erdos, Selkow] With high probability, after four iterations of stable coloring, each vertex of a random graph has a unique color, i.e., the $C_4^2$-type of each vertex is unique.

Thus, for almost all graphs, there is a linear time algorithm to canonize the graph, i.e., sort the vertices by their $C^2$ type, so that two graphs are isomorphic iff their canonical forms are equal.
The Good News: Upper Bounds

**Thm.** [Babai, Erdos, Selkow] With high probability, after four iterations of stable coloring, each vertex of a random graph has a unique color, i.e., the $C_4^2$-type of each vertex is unique.

Thus, for almost all graphs, there is a linear time algorithm to canonize the graph, i.e., sort the vertices by their $C_2^2$ type, so that two graphs are isomorphic iff their canonical forms are equal.

With high probability, $G \cong H$ iff $G \equiv^2_4 H$. 

Neil Immerman Towards Capturing Order-Independent P
**Thm.** [Babai, Erdos, Selkow] With high probability, after four iterations of stable coloring, each vertex of a random graph has a unique color, i.e., the $C_4^2$-type of each vertex is unique.

Thus, for almost all graphs, there is a linear time algorithm to canonize the graph, i.e., sort the vertices by their $C^2$ type, so that two graphs are isomorphic iff their canonical forms are equal.

With high probability, $G \cong H$ iff $G \equiv_4^2 H$.

Thus, Graph Isomorphism (GI) is linear time for random graphs.
**Thm.** [Babai, Erdos, Selkow] With high probability, after four iterations of stable coloring, each vertex of a random graph has a unique color, i.e., the $C_4^2$-type of each vertex is unique.

Thus, for almost all graphs, there is a linear time algorithm to canonize the graph, i.e., sort the vertices by their $C^2$ type, so that two graphs are isomorphic iff their canonical forms are equal.

With high probability, $G \cong H$ iff $G \equiv_4^2 H$.

Thus, Graph Isomorphism (GI) is linear time for random graphs.

In general the complexity of GI is unknown.
**Thm.** [Babai, Erdos, Selkow] With high probability, after four iterations of stable coloring, each vertex of a random graph has a unique color, i.e., the $C_4^2$-type of each vertex is unique.

Thus, for almost all graphs, there is a linear time algorithm to canonize the graph, i.e., sort the vertices by their $C^2$ type, so that two graphs are isomorphic iff their canonical forms are equal.

With high probability, $G \cong H$ iff $G \equiv_4^2 H$.

Thus, Graph Isomorphism (GI) is linear time for random graphs.

In general the complexity of GI is unknown.

**Thm.** [Babai, 2015] $\text{GI} \in \text{DTIME}[n^{\log^7 n}]$. (Before this it was only known that $\text{GI} \in \text{DTIME}[n^{\sqrt{n}}].$)
**Def.** Language $\mathcal{L}$ characterizes a graph $G$ iff for all graphs $H$,

$$G \equiv_{\mathcal{L}} H \iff G \cong H.$$
Def. Language $\mathcal{L}$ characterizes a graph $G$ iff for all graphs $H$,

$$G \equiv_{\mathcal{L}} H \iff G \cong H.$$ 

- $C^2$ characterizes almost all random graphs.
- $C^2$ characterizes all trees.
- $C^3$ characterizes all graphs of color class size 3.
Logics Characterizing Graphs

**Def.** Language $\mathcal{L}$ characterizes a graph $G$ iff for all graphs $H$,

$$G \equiv_{\mathcal{L}} H \iff G \cong H$$.

- $C^2$ characterizes almost all random graphs.
- $C^2$ characterizes all trees.
- $C^3$ characterizes all graphs of color class size 3.

**Thm.** We can test if $G \equiv_{C^k} H$ in FPC and $\text{DTIME}[n^k \log n]$. 
Def. Language $\mathcal{L}$ characterizes a graph $G$ iff for all graphs $H$, $G \equiv_\mathcal{L} H \iff G \cong H$.

- $C^2$ characterizes almost all random graphs.
- $C^2$ characterizes all trees.
- $C^3$ characterizes all graphs of color class size 3.

Thm. We can test if $G \equiv_{C^k} H$ in FPC and $\text{DTIME}[n^k \log n]$.

Cor. If $C^k$ characterizes all graphs in a class of graphs $\mathcal{G}$ that is closed under particularizing, then $\mathcal{G}$ admits $C^k$ canonization, and thus FPC captures OIP over $\mathcal{G}$. 

proof: Apply arbitrary FO$(LFP)$ formula to the canonical form of the input graph. □
Def. Language $\mathcal{L}$ characterizes a graph $G$ iff for all graphs $H$,

$$G \equiv_{\mathcal{L}} H \iff G \cong H.$$ 

- $C^2$ characterizes almost all random graphs.
- $C^2$ characterizes all trees.
- $C^3$ characterizes all graphs of color class size 3.

Thm. We can test if $G \equiv_{C^k} H$ in FPC and DTIME[$n^k \log n$].

Cor. If $C^k$ characterizes all graphs in a class of graphs $\mathcal{G}$ that is closed under particularizing, then $\mathcal{G}$ admits $C^k$ canonization, and thus FPC captures OIP over $\mathcal{G}$.

proof: Apply arbitrary FO(LFP) formula to the canonical form of the input graph.
Particularizing Means Uniquely Coloring Some Vertex

Towards Capturing Order-Independent P
Particularizing Means Uniquely Coloring Some Vertex

Neil Immerman
Towards Capturing Order-Independent P
Is FPC Equal to OIP?

▶ Is FPC Equal to OIP?

Does $C_4$ characterize all graphs?

If yes, then $FPC = OIP$ and for all graphs, $G \equiv C_4 H$. Thus, GI would be in $DTIME[\log n^4]$.

Thm. [CFI] No! A simple graph property (now called the CFI property) checkable in $DTIME[n]$, requires $v = \Omega(n)$ variables to express in $C_v$. Thus, $CFI \in OIP - FPC$. 

Neil Immerman Towards Capturing Order-Independent P
Is FPC Equal to OIP?

- Is FPC Equal to OIP?
- Does \( C^4 \) characterize all graphs?

Thm. [CFI] No!

A simple graph property (now called the CFI property) checkable in \( \text{DTIME} \left[ n \right] \), requires \( v = \Omega(n) \) variables to express in \( C^v \). Thus, CFI \( \in \text{OIP} - \text{FPC} \)
Is FPC Equal to OIP?

- Is FPC Equal to OIP?
- Does $C^4$ characterize all graphs?
- If yes, then $\text{FPC} = \text{OIP}$ and for all graphs, $G \approx H \iff G \equiv^{C^4} H$. Thus, GI would be in $\text{DTIME}[n^4 \log n]$. 

Thm. [CFI] No! A simple graph property (now called the CFI property) checkable in $\text{DTIME}[n]$, requires $v = \Omega(n)$ variables to express in $C^v$. Thus, CFI $\in \text{OIP} - \text{FPC}$.
Is FPC Equal to OIP?

- Is FPC Equal to OIP?
- Does $C^4$ characterize all graphs?
- If yes, then $FPC = OIP$ and for all graphs, $G \cong H \iff G \equiv_{C^4} H$.

Thus, GI would be in $\text{DTIME}[n^4 \log n]$.

**Thm.** [CFI] No!

A simple graph property (now called the CFI property) checkable in $\text{DTIME}[n]$, requires $v = \Omega(n)$ variables to express in $C^v$. Thus, $\text{CFI} \in \text{OIP} - \text{FPC}$
Proof of CFI Thm

Automorphisms of \( X \): switch an even number of \( (a_i b_i) \) pairs.

\[ \begin{align*}
\text{Automorphism:} & \quad (a_2 b_2) (a_3 b_3) (m_1 m_2) (m_3 m_4) \\
\text{Automorphism:} & \quad (a_1 b_1) (a_2 b_2) (m_1 m_4) (m_2 m_3)
\end{align*} \]
Proof of CFI Thm

CFI Gadget $X$: Each $m_i$ adjacent to an even number of $a_j$’s.
Proof of CFI Thm

CFI Gadget $X$: Each $m_i$ adjacent to an even number of $a_j$'s.

Automorphisms of $X$: switch an even number of $(a_i b_i)$ pairs.
Proof of CFI Thm

CFI Gadget $X$: Each $m_i$ adjacent to an even number of $a_j$'s.
Automorphisms of $X$: switch an even number of $(a_i b_i)$ pairs.

Automorphism: $(a_2 b_2)(a_3 b_3)(m_1 m_2)(m_3 m_4)$
Proof of CFI Thm

CFI Gadget $X$: Each $m_i$ adjacent to an even number of $a_j$’s.

Automorphisms of $X$: switch an even number of $(a_i b_i)$ pairs.

Automorphism: \[(a_1 b_1)(a_2 b_2)(m_1 m_4)(m_2 m_3)\]
Let $G_n$ be a regular, degree 3 graph with $O(n)$ vertices, color class size 1 and separator size $n$. 

If we remove any $n$ vertices from $G_n$, it still has a connected component with more than $|V_{G_n}|/2$ vertices.

Such regular degree 3 graphs with linear-size separators exist.

Color class size 1 means every vertex of $G_n$ has a unique color.

Let $X(G_n)$ be the result of replacing each vertex $v \in V_{G_n}$ by a copy of $X$ of $v$'s color.

Thus $X(G_n)$ has color class size 4.
Let $G_n$ be a regular, degree 3 graph with $O(n)$ vertices, color class size 1 and separator size $n$.

If we remove any $n$ vertices from $G_n$, it still has a connected component with more than $|V^{G_n}|/2$ vertices.
Let $G_n$ be a regular, degree 3 graph with $O(n)$ vertices, color class size 1 and separator size $n$.

If we remove any $n$ vertices from $G_n$, it still has a connected component with more than $|V^{G_n}|/2$ vertices.

Such regular degree 3 graphs with linear-size separators exist.
Let $G_n$ be a regular, degree 3 graph with $O(n)$ vertices, color class size 1 and separator size $n$.

If we remove any $n$ vertices from $G_n$, it still has a connected component with more than $|V^{G_n}|/2$ vertices.

Such regular degree 3 graphs with linear-size separators exist.

Color class size 1 means every vertex of $G_n$ has a unique color.
Let $G_n$ be a regular, degree 3 graph with $O(n)$ vertices, color class size 1 and separator size $n$.

If we remove any $n$ vertices from $G_n$, it still has a connected component with more than $|V^{G_n}|/2$ vertices.

Such regular degree 3 graphs with linear-size separators exist.

Color class size 1 means every vertex of $G_n$ has a unique color.

Let $X(G_n)$ be the result of replacing each vertex $v \in V^{G_n}$ by a copy of $X$ of $v$'s color.
Let $G_n$ be a regular, degree 3 graph with $O(n)$ vertices, color class size 1 and separator size $n$.

If we remove any $n$ vertices from $G_n$, it still has a connected component with more than $|V^{G_n}|/2$ vertices.

Such regular degree 3 graphs with linear-size separators exist.

Color class size 1 means every vertex of $G_n$ has a unique color.

Let $X(G_n)$ be the result of replacing each vertex $v \in V^{G_n}$ by a copy of $X$ of $v$'s color.

Thus $X(G_n)$ has color class size 4.
$X(G_n)$: replace each vertex $v \in V^{G_n}$ by a copy of $X$ of $v$’s color, connecting $a$ to $a$ and $b$ to $b$. 
\( \tilde{X}(G) \) is \( X(G) \) with any one edge pair flipped.
\( \tilde{X}(G) \) is \( X(G) \) with any one edge pair flipped.
Is it \( X(G_n) \) or \( \tilde{X}(G_n) \)?

**Prop.** Let \( X'(G_n) \) be \( X(G_n) \) with some number, \( m \), of the magenta edges flipped.

Then \( X'(G_n) \cong X(G_n) \) iff \( m \) is even and

\[
X'(G_n) \cong \tilde{X}(G_n) \text{ iff } m \text{ is odd.}
\]
Is it $X(G_n)$ or $\tilde{X}(G_n)$?

**Prop.** Let $X'(G_n)$ be $X(G_n)$ with some number, $m$, of the magenta edges flipped.

Then $X'(G_n) \cong X(G_n)$ iff $m$ is even and

$$X'(G_n) \cong \tilde{X}(G_n)$$

iff $m$ is odd.

**proof:** Using the automorphisms of $X$, we can move any two flips towards each other until they eliminate each other.
Prop. Let $X'(G_n)$ be $X(G_n)$ with some number, $m$, of the magenta edges flipped.

Then $X'(G_n) \cong X(G_n)$ iff $m$ is even and

$$X'(G_n) \cong \tilde{X}(G_n)$$

iff $m$ is odd.
Is it $X(G_n)$ or $\tilde{X}(G_n)$?

**Prop.** Let $X'(G_n)$ be $X(G_n)$ with some number, $m$, of the magenta edges flipped.

Then $X'(G_n) \cong X(G_n)$ iff $m$ is even and

$$X'(G_n) \cong \tilde{X}(G_n)$$

iff $m$ is odd.

**proof:** Using the automorphisms of $X$, we can move any two flips towards each other until they eliminate each other. □
$\tilde{X}(G_n)$
$\tilde{X}(G_n)$
Every one of the $m_i$'s is connected to an even number of $a_j$'s.
Every one of the \( m_i \)'s is connected to an odd number of \( a_j \)'s.
Def. $\text{CFI} = \{(X'(G) \mid X'(G) \cong X(G)\}$ for $G$ is connected, reg. deg. 3, $\text{cc}(G) = 1$. 
The CFI Problem

Def. \( \text{CFI} = \{ (X'(G) \ | \ X'(G) \cong X(G) \} \) for \( G \) is connected, reg. deg. 3, \( cc(G) = 1 \).

Prop. \( \text{CFI} \in \text{DTIME}[n] \).
**The CFI Problem**

**Def.** $\text{CFI} = \{(X'(G) \mid X'(G) \cong X(G)\}$ for $G$ is connected, reg. deg. 3, $\text{cc}(G) = 1$.

**Prop.** $\text{CFI} \in \text{DTIME}[n]$.

**proof** Use the ordering to label boundary pairs $a_i, b_i$ when $a_i \leq b_i$. Then count the number, $m$, of flips of vertices and edges mod 2. $X'(G) \in \text{CFI}$ iff $m$ is even. □
$\tilde{X}(G_n)$
Thm. \[ \text{CFI} \in \text{OIP} - \text{FPC}. \]
Thm.  \( \text{CFI} \in \text{OIP} - \text{FPC}. \)

proof We show that \( X(G_n) \equiv_{C^n} \tilde{X}(G_n) \).
**Thm.** CFI $\in$ OIP $-$ FPC.

**proof** We show that $X(G_n) \equiv^n \tilde{X}(G_n)$.

Counting doesn’t help since $\text{cc}(X(G_n)) = 4$. Suffices to show that $X(G_n) \sim^n \tilde{X}(G_n)$. 
Thm. \[ \text{CFI} \in \text{OIP} - \text{FPC}. \]

proof We show that \( X(G_n) \equiv^c \tilde{X}(G_n) \).

Counting doesn’t help since \( \text{cc}(X(G_n)) = 4 \). Suffices to show that \( X(G_n) \sim^n \tilde{X}(G_n) \).

Initially no pebbles on the board, **Samson** places \( x_1 \) on \( X(v) \) in one of the two graphs. Note that the largest connected component \( C_1 \) of \( G - \{v\} \) includes over half the vertices of \( G \). **Delilah** moves the flip into \( C_1 \). If she removes the flip, then the two graphs are isomorphic. **Delilah** answers according to this isomorphism.
Thm. $\text{CFI} \in \text{OIP} - \text{FPC}$. 

proof We show that $X(G_n) \equiv_C X(G_n)$. 

Counting doesn’t help since $\text{cc}(X(G_n)) = 4$. Suffices to show that $X(G_n) \sim X(G_n)$. 

Initially no pebbles on the board, Samson places $x_1$ on $X(v)$ in one of the two graphs. Note that the largest connected component $C_1$ of $G - \{v\}$ includes over half the vertices of $G$. Delilah moves the flip into $C_1$. If she removes the flip, then the two graphs are isomorphic. Delilah answers according to this isomorphism. 

Inductively, after step $m$, Delilah has not yet lost, so there is an isomorphism from chosen points in $X(G_n)$ to chosen points in $X(G_n)$ which extends to an isomorphism of the whole graphs in which a flip in $G_n$ in $C_m$ has been removed.
Inductively, after step $m$, **Delilah** has not yet lost, so there is an isomorphism from chosen points in $X(G_n)$ to chosen points in $\tilde{X}(G_n)$ which extends to an isomorphism of the whole graphs in which a flip in $\tilde{G}_n$ in $C_m$ has been removed.

---

Neil Immerman
Towards Capturing Order-Independent P
Inductively, after step $m$, Delilah has not yet lost, so there is an isomorphism from chosen points in $X(G_n)$ to chosen points in $\tilde{X}(G_n)$ which extends to an isomorphism of the whole graphs in which a flip in $G_n$ in $C_m$ has been removed.

Samson picks up the $x_i$ pebbles and places one on some $X(v)$. Note that $C_m$ and $C_{m+1}$ both contain over half the vertices of $G_n$.

Thus they have some vertex $w \in C_m \cap C_{m+1}$.
Inductively, after step $m$, Delilah has not yet lost, so there is an isomorphism from chosen points in $X(G_n)$ to chosen points in $\tilde{X}(G_n)$ which extends to an isomorphism of the whole graphs in which a flip in $\tilde{G}_n$ in $C_m$ has been removed.

Samson picks up the $x_i$ pebbles and places one on some $X(v)$. Note that $C_m$ and $C_{m+1}$ both contain over half the vertices of $G_n$.

Thus they have some vertex $w \in C_m \cap C_{m+1}$.

Delilah mentally moves the flip to $X(w)$. She then answers according to the isomorphism from $X(G_n)$ to $\tilde{X}(G_n)$ where that flip in $X(w)$ has been removed.
Inductively, after step $m$, **Delilah** has not yet lost, so there is an isomorphism from chosen points in $X(G_n)$ to chosen points in $\tilde{X}(G_n)$ which extends to an isomorphism of the whole graphs in which a flip in $\tilde{G}_n$ in $C_m$ has been removed.

**Samson** picks up the $x_i$ pebbles and places one on some $X(v)$. Note that $C_m$ and $C_{m+1}$ both contain over half the vertices of $G_n$.

Thus they have some vertex $w \in C_m \cap C_{m+1}$.

**Delilah** mentally moves the flip to $X(w)$. She then answers according to the isomorphism from $X(G_n)$ to $\tilde{X}(G_n)$ where that flip in $X(w)$ has been removed.

Thus **Delilah** never loses. □
We have shown that the linear-time CFI problem is in $\text{OIP} - \text{FPC}$.

**Cor.** $\Omega(n)$ variables are needed to characterize graphs.
Recent Developments: FPC is Surprisingly Powerful

Martin Grohe has shown that many classes of graphs are characterized by $C^k$ for some $k$. This includes planer graphs, graphs of bounded genus, graphs of bounded tree width and culminating in

Thm. [Grohe] Any class $G$ of graphs that excludes some minor is characterized by $C^k$ for some fixed $k$.

Thus, $FPC$ captures $OIP$ on $G$. Thus, for graphs from $G$, graph isomorphism and canonization are in $P$.

For $G$, $H \in G$, $G \cong H$ iff $G \equiv C^k H$.

Thm. [Anderson, Dawar and Holm] Linear Programming is in $FPC$.
Martin Grohe has shown that many classes of graphs are characterized by $C^k$ for some $k$. This includes planer graphs, graphs of bounded genus, graphs of bounded tree width and culminating in

**Thm.** [Grohe] Any class $\mathcal{G}$ of graphs that excludes some minor is characterized by $C^k$ for some fixed $k$. 

Towards Capturing Order-Independent P
Martin Grohe has shown that many classes of graphs are characterized by $C^k$ for some $k$. This includes planer graphs, graphs of bounded genus, graphs of bounded tree width and culminating in

**Thm.** [Grohe] Any class $\mathcal{G}$ of graphs that excludes some minor is characterized by $C^k$ for some fixed $k$. Thus,

- FPC captures OIP on $\mathcal{G}$. Thus, for graphs from $\mathcal{G}$, graph isomorphism and canonization are in P.
- For $G, H \in \mathcal{G}$, $G \cong H$ iff $G \equiv_{C^k} H$. 

Thm. [Anderson, Dawar and Holm] Linear Programming is in FPC.
Martin Grohe has shown that many classes of graphs are characterized by $C^k$ for some $k$. This includes planar graphs, graphs of bounded genus, graphs of bounded tree width and culminating in

**Thm.** [Grohe] Any class $\mathcal{G}$ of graphs that excludes some minor is characterized by $C^k$ for some fixed $k$. Thus,

- FPC captures OIP on $\mathcal{G}$. Thus, for graphs from $\mathcal{G}$, graph isomorphism and canonization are in P.

- For $G, H \in \mathcal{G}$, $G \cong H$ iff $G \equiv_{C^k} H$.

**Thm.** [Anderson, Dawar and Holm] Linear Programming is in FPC.
Two other languages are candidates for capturing OIP:

- Choiceless Polynomial Time (CPT) [Blass and Gurevich] Compute using sets of sets of sets, etc., where instead of choosing the first vertex, we consider the set of all such choices, keeping the total size of all sets polynomial.
- Rank Logic [Dawar, Grohe, Holm, and Laubner] Compute the rank of matrices expressed in an unordered setting.

CFI is expressible in CPT and in Rank Logic, thus these are strict extensions of FPC.

What I want: more natural extension to FPC that adds group theory and characterizes graphs using $O(\log n)$ variables.
Two other languages are candidates for capturing OIP:

- **Choiceless Polynomial Time (CPT)** [Blass and Gurevich]
  Compute using sets of sets of sets, etc., where instead of choosing the first vertex, we consider the set of all such choices, keeping the total size of all sets polynomial.
Going Beyond FPC

Two other languages are candidates for capturing OIP:

- **Choiceless Polynomial Time (CPT) [Blass and Gurevich]**
  Compute using sets of sets of sets, etc., where instead of choosing the first vertex, we consider the set of all such choices, keeping the total size of all sets polynomial.

- **Rank Logic [Dawar, Grohe, Holm, and Laubner]**
  Compute the rank of matrices expressed in an unordered setting.

What I want: more natural extension to FPC that adds group theory and characterizes graphs using $O(\log n)$ variables.
Two other languages are candidates for capturing OIP:

- **Choiceless Polynomial Time (CPT)** [Blass and Gurevich]
  Compute using sets of sets of sets, etc., where instead of choosing the first vertex, we consider the set of all such choices, keeping the total size of all sets polynomial.

- **Rank Logic** [Dawar, Grohe, Holm, and Laubner]
  Compute the rank of matrices expressed in an unordered setting.

CFI is expressible in CPT and in Rank Logic, thus these are strict extensions of FPC.
Going Beyond FPC

Two other languages are candidates for capturing OIP:

▶ Choiceless Polynomial Time (CPT) [Blass and Gurevich] Compute using sets of sets of sets, etc., where instead of choosing the first vertex, we consider the set of all such choices, keeping the total size of all sets polynomial.

▶ Rank Logic [Dawar, Grohe, Holm, and Laubner] Compute the rank of matrices expressed in an unordered setting.

CFI is expressible in CPT and in Rank Logic, thus these are strict extensions of FPC.

What I want: more natural extension to FPC that adds group theory and characterizes graphs using $O(\log n)$ variables.
Arithmetic Hierarchy

FO(N) r.e. complete

Halt

co-r.e. complete

Halt

FO∃(N) r.e.

FO∀(N) co-r.e.

Recursive

Primitive Recursive

SO[2^n]

EXPTIME

FO[2^n]

QSAT PSPACE complete

SO[n]

PSPACE

PTIME Hierarchy

SO

NP complete

SAT

co-NP complete

NP

co-NP

NP \cap co-NP

P complete

P

Horn-SAT

“truly feasible”

NC

FO[log n]

AC^1

FO(CFL)

sAC^1

FO(TC)

2SAT NL comp.

NL

FO(DTC)

2COLOR L comp.

L

FO(REGULAR)

NC^1

FO(COUNT)

ThC^0

FO

LOGTIME Hierarchy

AC^0

Neil Immerman
Towards Capturing Order-Independent P