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				**Answer**
$q_1\ q_2\ \cdots\ q_n$	$\mapsto$	**Computation**	$\mapsto$	$a_1\ a_2\ \cdots\ a_i\ \cdots\ a_m$

Descriptive Complexity

Query $\mapsto$ **Computation** $\mapsto$ **Answer**

$q_1\ q_2\ \cdots\ q_n$ $\mapsto$ **Computation** $\mapsto$ $a_1\ a_2\ \cdots\ a_i\ \cdots\ a_m$

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

Descriptive Complexity

Query				**Answer**
$q_1 \; q_2 \; \cdots \; q_n$	$\mapsto$	**Computation**	$\mapsto$	$a_1 \; a_2 \; \cdots \; a_i \; \cdots \; a_m$
				$\cdots \; S \; \cdots$

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

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

Descriptive Complexity

Query				**Answer**
$q_1\ q_2\ \cdots\ q_n$	$\mapsto$	**Computation**	$\mapsto$	$a_1\ a_2\ \cdots\ a_i\ \cdots\ a_m$
				$\cdots\ S\ \cdots$

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

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

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

Descriptive Complexity

Query				**Answer**
$q_1 \; q_2 \; \cdots \; q_n$	$\mapsto$	**Computation**	$\mapsto$	$a_1 \; a_2 \; \cdots \; a_i \; \cdots \; a_m$
				$\cdots \; S \; \cdots$

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

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

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

There is a **computable isomorphism** between these two approaches.

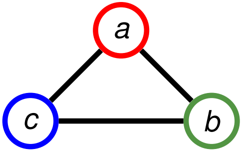
Think of the Input as a Finite Logical Structure

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

Colored

Graph

H

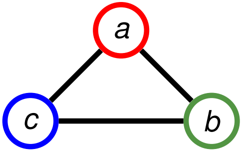


Think of the Input as a Finite Logical Structure

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

Colored Graph $\quad E^H = \{(a,b),(b,a),(b,c),(c,b),(c,a),(a,c)\}$

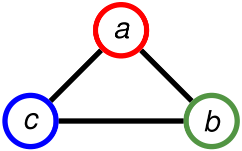
H



Think of the Input as a Finite Logical Structure

H		$=$	$(\{a,b,c\}, \leq, E^H, R^H, G^H, B^H)$
Colored	E^H	$=$	$\{(a,b),(b,a),(b,c),(c,b),(c,a),(a,c)\}$
Graph	R^H	$=$	$\{a\}$
	G^H	$=$	$\{b\}$
	B^H	$=$	$\{c\}$

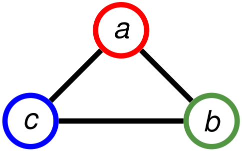
H



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	$=$	$\{c\}$

H



First-Order Logic

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

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

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

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

First-Order Logic

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

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

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

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

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:	$\wedge, \vee, \neg$
quantifiers:	$\forall, \exists$
numeric symbols:	$=, \leq, +, \times, min, max$

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

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

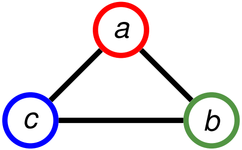
$$\gamma \equiv \forall x\, ((\forall y\; x \leq y) \rightarrow R(x))$$

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

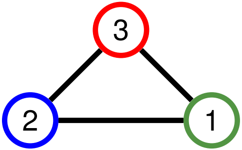
It is **easy** to test if input, H, satisfies α $(H \models \alpha)$.

First-Order Logic

$H \quad a \leq b \leq c$



$G \quad 1 \leq 2 \leq 3$



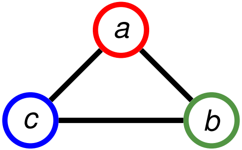
$$\alpha \equiv \forall x \exists y \, E(x, y)$$

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

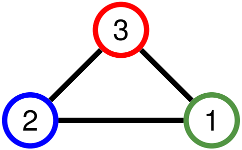
$$\gamma \equiv \forall x \, ((\forall y \; x \leq y) \rightarrow R(x))$$

First-Order Logic

H $\quad a \le b \le c$



G $\quad 1 \le 2 \le 3$



$H \models \alpha \wedge \beta \wedge \gamma$

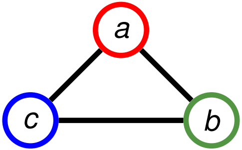
$$\alpha \equiv \forall x \exists y\, E(x,y)$$

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

$$\gamma \equiv \forall x\, ((\forall y\; x \le y) \;\rightarrow\; R(x))$$

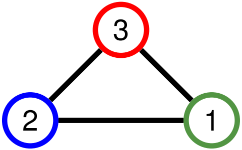
First-Order Logic

$H \qquad a \leq b \leq c$



$H \models \alpha \wedge \beta \wedge \gamma$

$G \qquad 1 \leq 2 \leq 3$



$G \models \alpha \wedge \beta \wedge \neg\gamma$

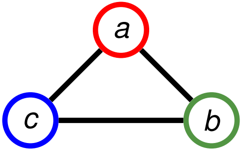
$$\alpha \equiv \forall x \exists y \, E(x, y)$$

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

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

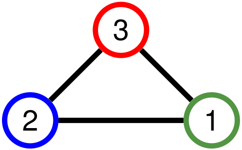
First-Order Logic

$H \quad a \leq b \leq c$



$H \models \alpha \wedge \beta \wedge \gamma$

$G \quad 1 \leq 2 \leq 3$



$G \models \alpha \wedge \beta \wedge \neg\gamma$

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

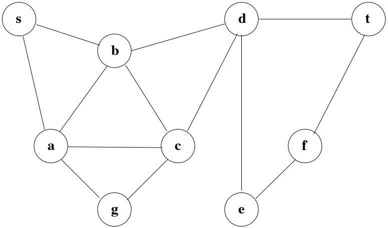
$$\beta \equiv \forall xy\, (\neg E(x,x) \wedge (E(x,y) \rightarrow E(y,x)))$$

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

α and β are **order independent**; γ is **order dependent**

Second-Order Logic: FO plus Relation Variables

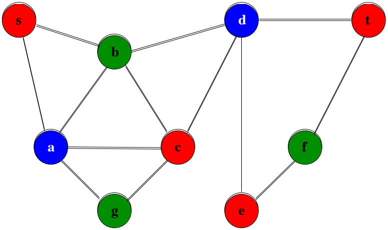
$$\Phi_{\text{3color}} \equiv \exists R^1 G^1 B^1 \forall x\, y\, ((R(x) \vee G(x) \vee B(x)) \wedge (E(x,y) \rightarrow (\neg(R(x) \wedge R(y)) \wedge \neg(G(x) \wedge G(y)) \wedge \neg(B(x) \wedge B(y)))))$$

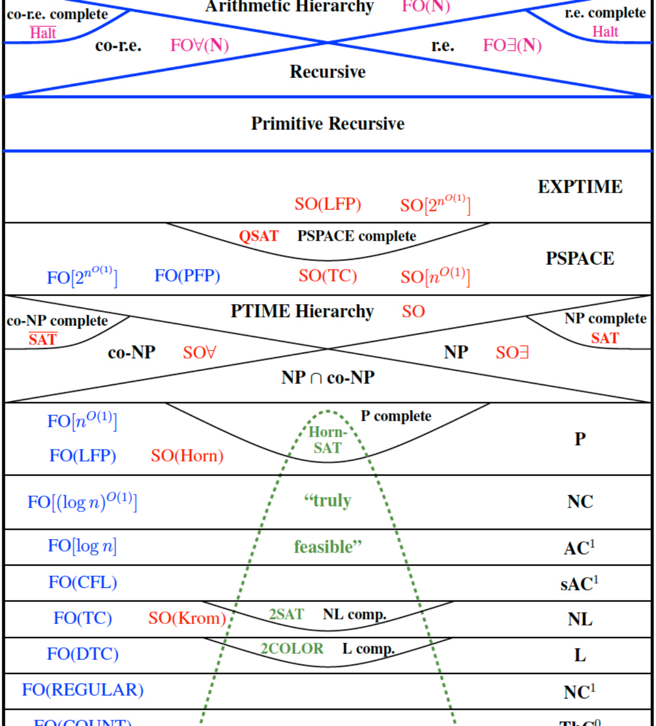


Second-Order Logic: FO plus Relation Variables

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

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





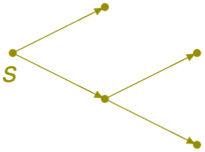
Arithmetic Hierarchy
FO(N)
co-r.e. complete
Halt
r.e. complete
Halt
co-r.e.
FO∀(N)
r.e.
FO∃(N)
Recursive
Primitive Recursive
EXPTIME
SO(LFP)
SO[$2^{n^{O(1)}}$]
QSAT
PSPACE complete
PSPACE
FO[$2^{n^{O(1)}}$]
FO(PFP)
SO(TC)
SO[$n^{O(1)}$]
PTIME Hierarchy
SO
co-NP complete
SAT
NP complete
SAT
co-NP
SO∀
NP
SO∃
NP ∩ co-NP
FO[$n^{O(1)}$]
P complete
Horn-SAT
FO(LFP)
SO(Horn)
P
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

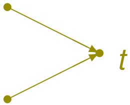
Inductive Definitions and Least Fixed Point

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

Inductive Definitions and Least Fixed Point

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

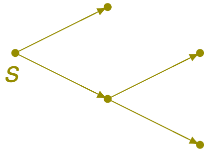


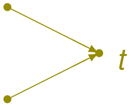


Inductive Definitions and Least Fixed Point

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

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



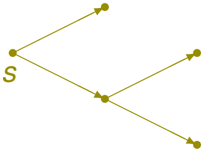


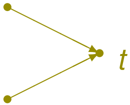
Inductive Definitions and Least Fixed Point

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

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

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





Inductive Definitions and Least Fixed Point

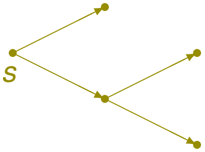
$$E^\star(x,y) \;\stackrel{\text{def}}{=}\; x = y \;\vee\; E(x,y) \;\vee\; \exists z (E^\star(x,z) \wedge E^\star(z,y))$$

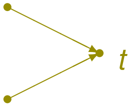
$$\varphi_{tc}(R,x,y) \;\equiv\; x = y \;\vee\; E(x,y) \;\vee\; \exists z (R(x,z) \wedge R(z,y))$$

$$\varphi_{tc}^G : \text{binRel}(G) \;\to\; \text{binRel}(G)$$

monotone $\quad R \subseteq S \quad \Rightarrow \quad \varphi_{tc}^G(R) \subseteq \varphi_{tc}^G(S)$

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





Inductive Definitions and Least Fixed Point

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

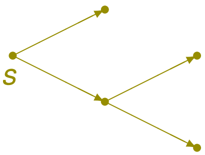
$$\varphi_{tc}(R,x,y) \equiv x = y \;\vee\; E(x,y) \;\vee\; \exists z(R(x,z) \wedge R(z,y))$$

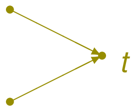
$$\varphi_{tc}^G : \text{binRel}(G) \to \text{binRel}(G)$$

monotone $\quad R \subseteq S \quad \Rightarrow \quad \varphi_{tc}^G(R) \subseteq \varphi_{tc}^G(S)$

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

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





Inductive Definitions and Least Fixed Point

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

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

$$\varphi_{tc}^{G} : \text{binRel}(G) \to \text{binRel}(G)$$

monotone $\quad R \subseteq S \quad \Rightarrow \quad \varphi_{tc}^{G}(R) \subseteq \varphi_{tc}^{G}(S)$

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

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

LFP is a Polynomial Iteration Operator

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

$\mathrm{FO}[n^{O(1)}]$ means for graphs with n vertices, the formula φ_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. $\mathrm{P} = \mathrm{FO(LFP)} = \mathrm{FO}[n^{O(1)}]$

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

$\mathrm{FO}[n^{O(1)}]$ means for graphs with n vertices, the formula φ_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. $\mathrm{P} = \mathrm{FO(LFP)} = \mathrm{FO}[n^{O(1)}]$

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

$\mathrm{FO}[n^{O(1)}]$ means for graphs with n vertices, the formula φ_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. $\mathrm{P} = \mathrm{FO(LFP)} = \mathrm{FO}[n^{O(1)}]$

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

$\mathrm{FO}[n^{O(1)}]$ means for graphs with n vertices, the formula φ_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. $\mathrm{P} = \mathrm{FO(LFP)} = \mathrm{FO}[n^{O(1)}]$

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

$\mathrm{FO}[n^{O(1)}]$ means for graphs with n vertices, the formula φ_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.

LFP is a Polynomial Iteration Operator

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

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

$\mathrm{FO}[n^{O(1)}]$ means for graphs with n vertices, the formula φ_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. $\mathrm{P} = \mathrm{FO(LFP)} = \mathrm{FO}[n^{O(1)}]$

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

$\mathrm{FO}[n^{O(1)}]$ means for graphs with n vertices, the formula φ_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-Independent P (OIP)

$\mathrm{FO(LFP)} \; = \; \mathrm{P}$

$\mathrm{FO}_{(\mathrm{wo}\leq)}\mathrm{(LFP)} \; \subseteq$ **OIP**

Want to Capture Order-Independent P (OIP)

$\mathrm{FO(LFP)} \;=\; \mathrm{P}$

$\mathrm{FO(wo\leq)(LFP)} \;\subseteq\;$ **OIP**

$\mathsf{EVEN} \stackrel{\text{def}}{=} \{G \mid |V^G| \equiv 0 \pmod 2\}$

Want to Capture Order-Independent P (OIP)

$\mathrm{FO(LFP)} \;=\; \mathrm{P}$

$\mathrm{FO(wo{\le})(LFP)} \;\subseteq\;$ **OIP**

$\text{EVEN} \;\stackrel{\text{def}}{=}\; \{G \mid |V^G| \equiv 0 \pmod 2\}$

$\text{EVEN} \in$ **OIP** $- \mathrm{FO(wo{\le})(LFP)}$.

Want to Capture Order-Independent P (OIP)

$\mathrm{FO(LFP)} \ = \ \mathrm{P}$

$\mathrm{FO(wo\leq)(LFP)} \ \subseteq$ **OIP**

EVEN $\stackrel{\mathrm{def}}{=} \{G \mid |V^G| \equiv 0 \pmod 2\}$

EVEN $\in$ **OIP** $- \mathrm{FO(wo\leq)(LFP)}$.

Thus, $\quad \mathrm{FO(wo\leq)(LFP)} \ \subsetneq$ **OIP**

Want to Capture Order-Independent P (OIP)

$\mathrm{FO(LFP)} \;=\; \mathrm{P}$

$\mathrm{FO(wo{\leq})(LFP)} \;\subseteq\;$ **OIP**

EVEN $\stackrel{\text{def}}{=} \{G \mid |V^G| \equiv 0 \pmod 2\}$

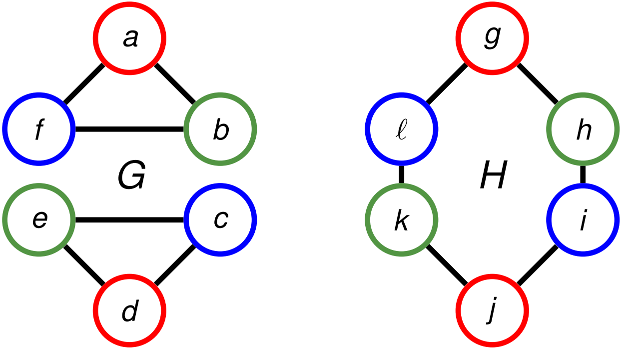
EVEN $\in$ **OIP** $-\ \mathrm{FO(wo{\leq})(LFP)}$.

Thus, $\quad \mathrm{FO(wo{\leq})(LFP)} \;\subsetneq\;$ **OIP**

How do we prove EVEN $\notin \mathrm{FO(wo{\leq})(LFP)}$?

Ehrenfeucht-Fraïssé Game

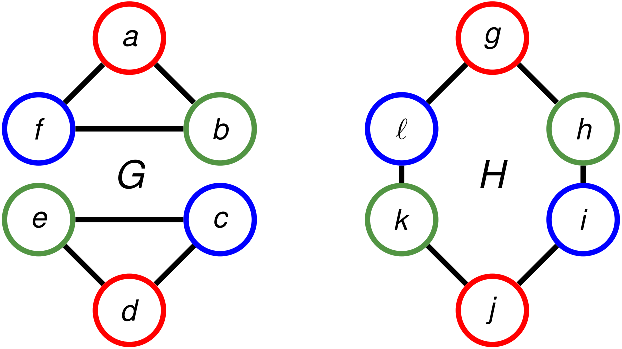
$\mathcal{G}_m^k(G, H)$ m moves, k pebbles, 2 players



Ehrenfeucht-Fraïssé Game

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

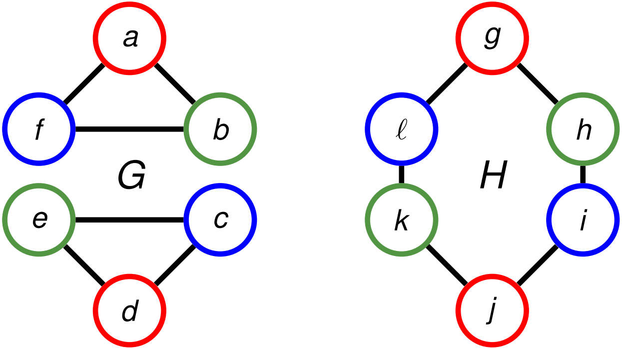
Samson: show a difference.



Ehrenfeucht-Fraïssé Game

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

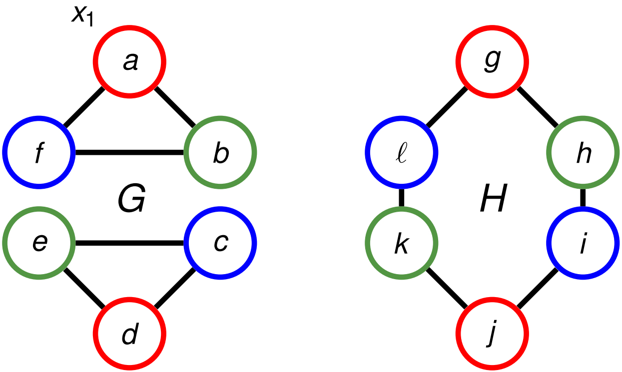
Samson: show a difference.   **Delilah**: preserve isomorphism.



Ehrenfeucht-Fraïssé Game

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

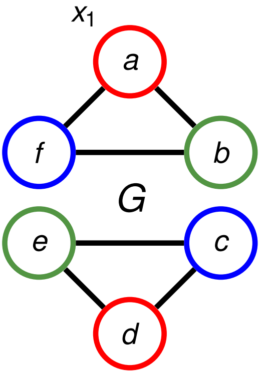
Samson: show a difference.   **Delilah**: preserve isomorphism.



Ehrenfeucht-Fraïssé Game

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

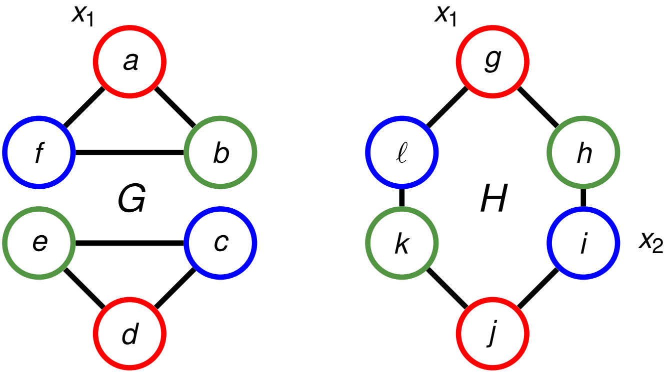
Samson: show a difference. **Delilah**: preserve isomorphism.



Ehrenfeucht-Fraïssé Game

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

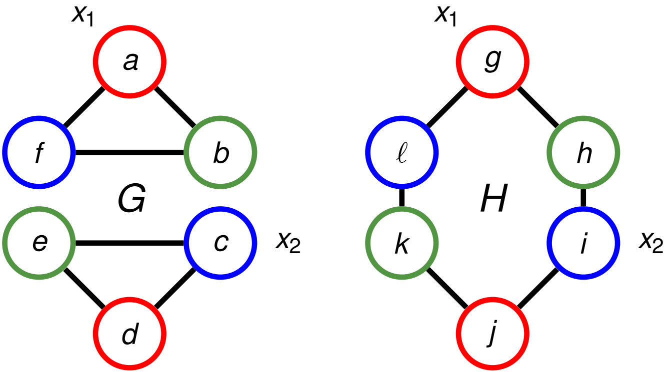
Samson: show a difference. **Delilah**: preserve isomorphism.



Ehrenfeucht-Fraïssé Game

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

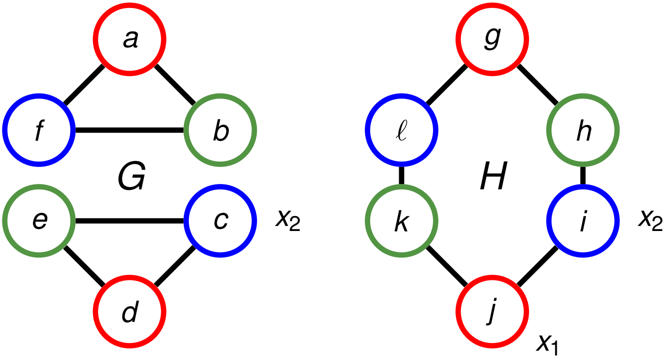
Samson: show a difference.   **Delilah**: preserve isomorphism.



Ehrenfeucht-Fraïssé Game

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

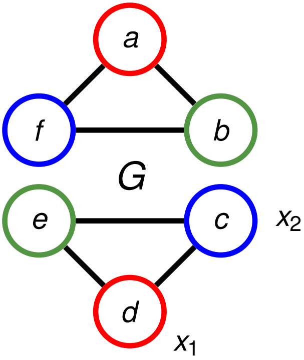
Samson: show a difference. **Delilah**: preserve isomorphism.



Ehrenfeucht-Fraïssé Game

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

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

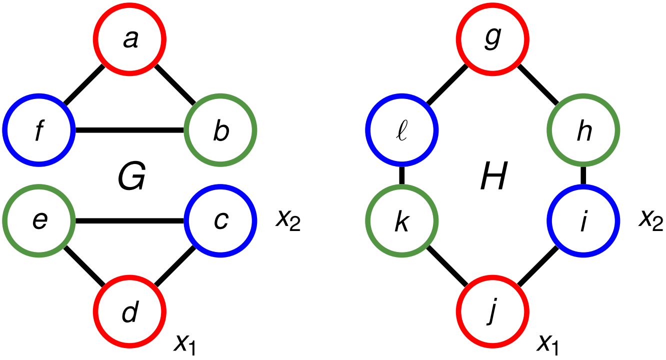


Ehrenfeucht-Fraïssé Game

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

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

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

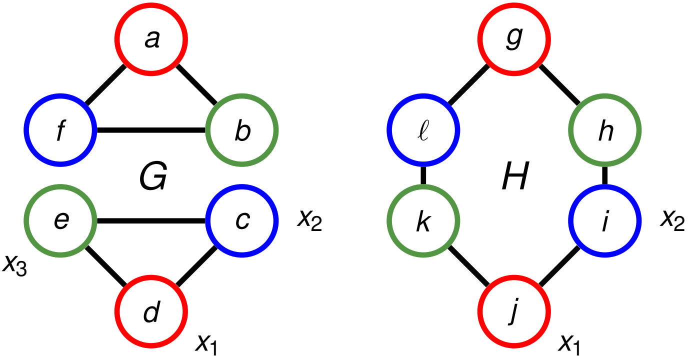


Ehrenfeucht-Fraïssé Game

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

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

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

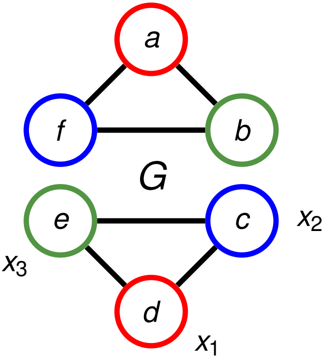


Ehrenfeucht-Fraïssé Game

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

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

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

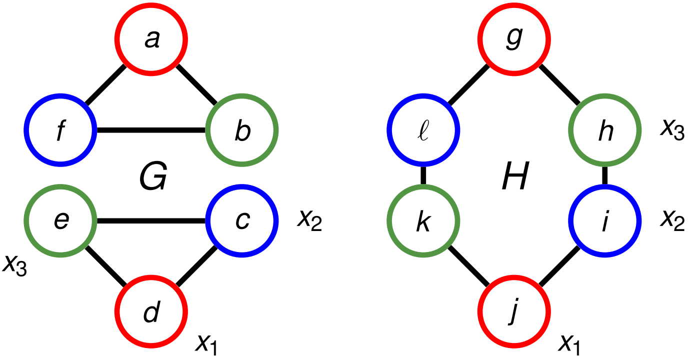


Ehrenfeucht-Fraïssé Game

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

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

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

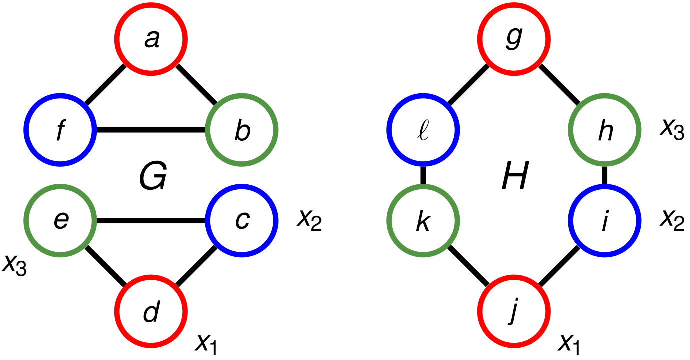


Ehrenfeucht-Fraïssé Game

$\mathcal{G}_m^k(G, H)$ 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 $\mathcal{G}_3^3(G, H)$.



Fundamental Thm of Ehrenfeucht-Fraïssé Games

Notation: $G \sim_m^k H$ means that **Delilah** has a winning strategy for $\mathcal{G}_m^k(G, H)$.

Fundamental Thm of Ehrenfeucht-Fraïssé Games

Notation: $G \sim^k_m H$ means that **Delilah** has a winning strategy for $\mathcal{G}^k_m(G, H)$.

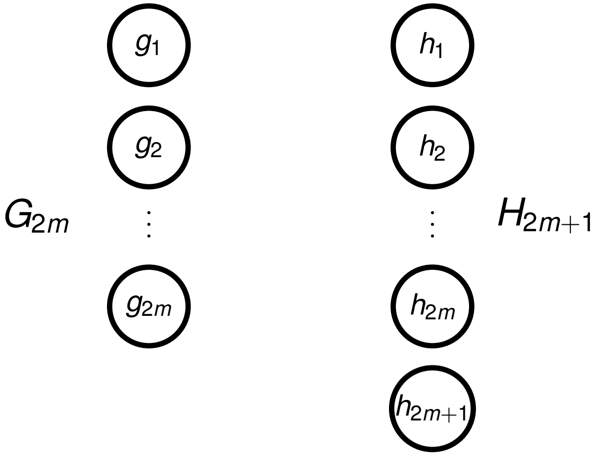
Thm. **D** has a winning strategy on the m-move, k-pebble game on G, H iff G and H agree on all formulas using k variables and quantifier depth m.

$$G \sim^k_m H \quad \Leftrightarrow \quad G \equiv^k_m H$$

Thm. **EVEN** requires $n+1$ variables without ordering.
Thus **EVEN** $\notin \mathrm{FO(wo{\leq})(LFP)}$.

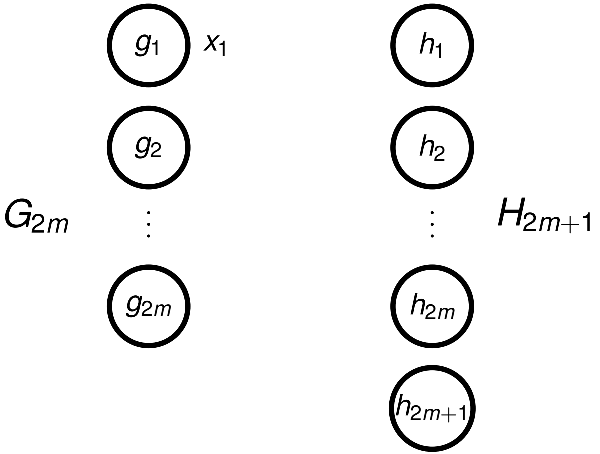
Thm. **EVEN** requires $n+1$ variables without ordering.
Thus **EVEN** $\notin \mathrm{FO(wo\leq)(LFP)}$.

proof:



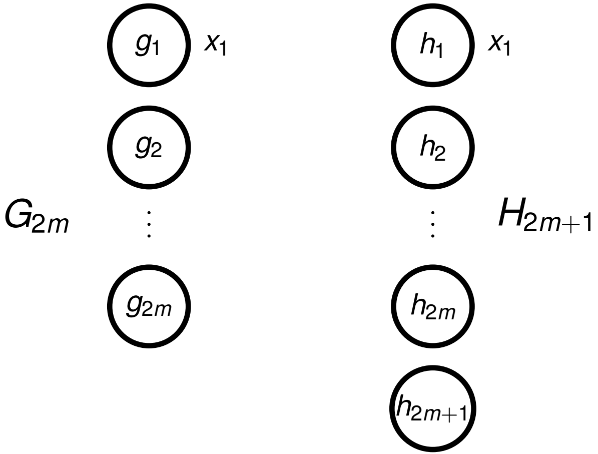
Thm. **EVEN** requires $n+1$ variables without ordering.
Thus **EVEN** $\not\in \mathrm{FO(wo{\leq})(LFP)}$.

proof:



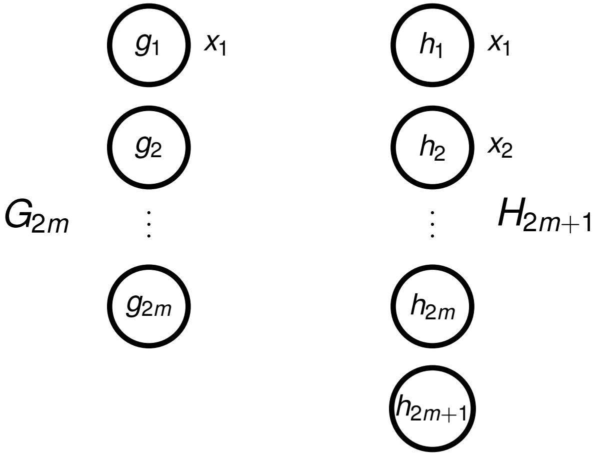
Thm. **EVEN** requires $n+1$ variables without ordering.
Thus **EVEN** $\notin \mathrm{FO(wo{\leq})(LFP)}$.

proof:



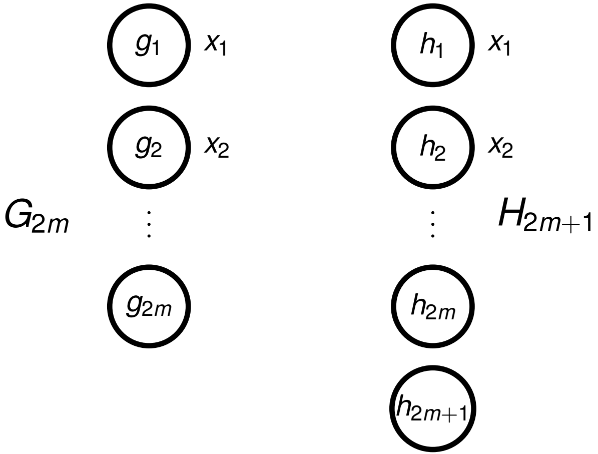
Thm. **EVEN** requires $n+1$ variables without ordering.
Thus **EVEN** $\notin \mathrm{FO(wo\leq)(LFP)}$.

proof:



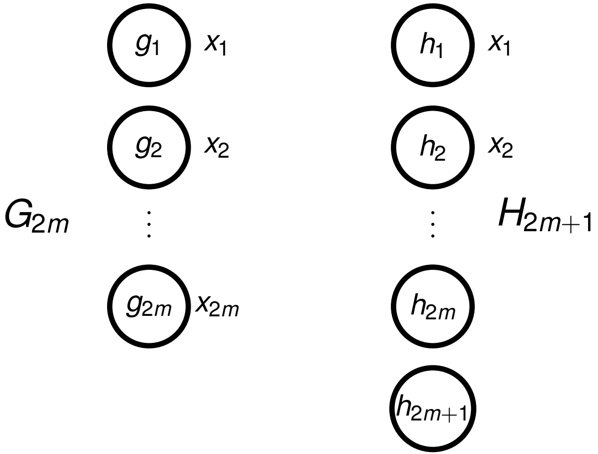
Thm. **EVEN** requires $n+1$ variables without ordering.
Thus **EVEN** $\notin \mathrm{FO(wo\leq)(LFP)}$.

proof:



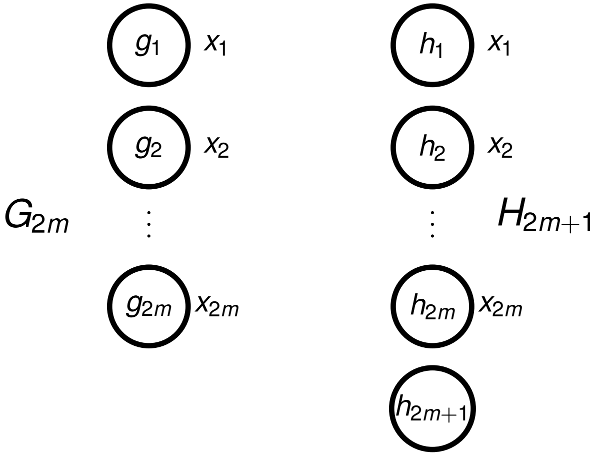
Thm. **EVEN** requires $n + 1$ variables without ordering.
Thus **EVEN** $\notin \mathrm{FO(wo\leq)(LFP)}$.

proof:



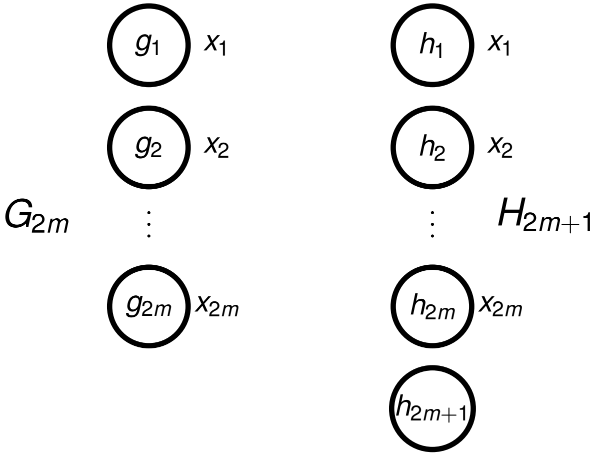
Thm. **EVEN** requires $n+1$ variables without ordering.
Thus **EVEN** $\notin \mathrm{FO(wo{\leq})(LFP)}$.

proof:



Thm. **EVEN** requires $n+1$ variables without ordering. Thus **EVEN** $\notin \mathrm{FO(wo\le)(LFP)}$.

proof:



$$G_{2m} \sim^{2m} H_{2m+1}$$

□

Add Counting to FO Logic

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

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: $\#x(\varphi(x))$.

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: $\#x(\varphi(x))$.

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

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: $\#x(\varphi(x))$.

$$\mathrm{EVEN} \quad \equiv \quad \exists i\,(\mathrm{Plus}(i, i, \#x(x = x)))$$

Let $C^k \stackrel{\mathrm{def}}{=} \mathrm{FO}^k(\mathrm{COUNT})$; $\mathrm{FPC} \stackrel{\mathrm{def}}{=} \mathrm{FO}(\mathrm{LFP}, \mathrm{COUNT})$.

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: $\#x(\varphi(x))$.

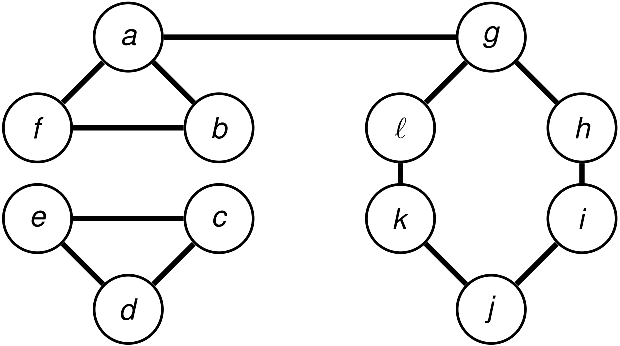
$$\text{EVEN} \quad \equiv \quad \exists i \, (\text{Plus}(i, i, \#x(x = x)))$$

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

$$\mathrm{FO}(\mathrm{wo}\leq)(\mathrm{LFP}) \subsetneq \mathrm{FPC} \subseteq \textbf{OIP}$$

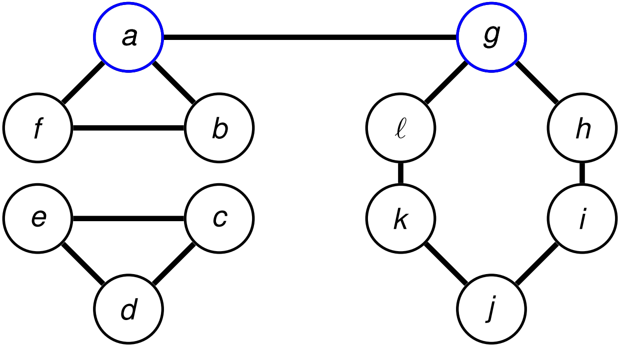
Stable Coloring of Vertices

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



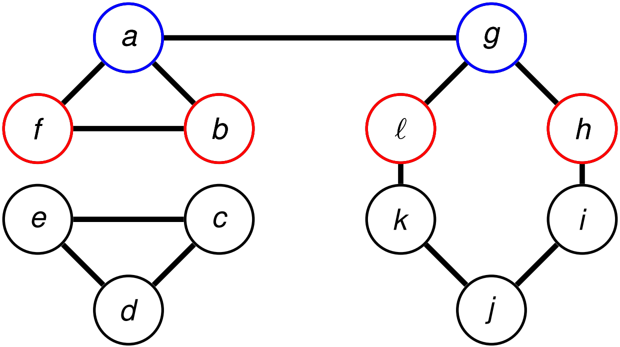
Stable Coloring of Vertices

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



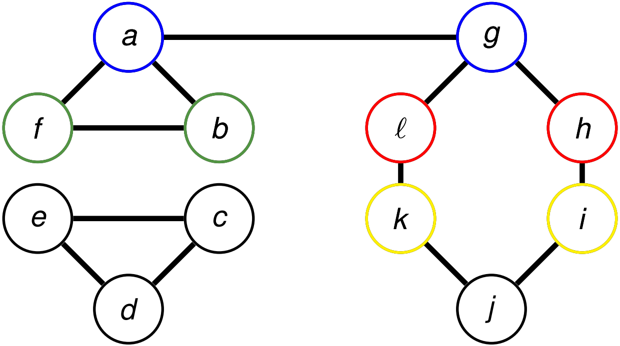
Stable Coloring of Vertices

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



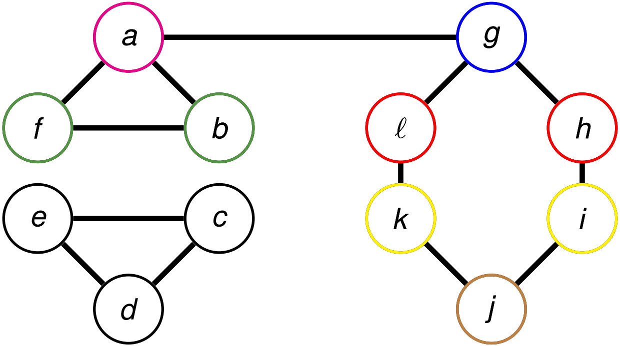
Stable Coloring of Vertices

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



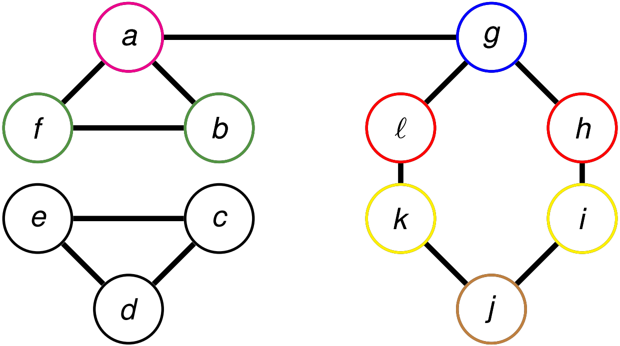
Stable Coloring of Vertices

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 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.

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 type, so that two graphs are isomorphic iff their canonical forms are equal.

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

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.

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

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

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^2_4-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 $G \equiv^2_4 H$.

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

In general the complexity of GI is unknown.

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^2_4-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, $\quad G \cong H$ iff $G \equiv^2_4 H$.

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

In general the complexity of GI is unknown.

Thm. [Babai, 2015] $\quad \mathrm{GI} \in \mathrm{DTIME}[n^{\log^7 n}]$. (Before this it was only known that $\mathrm{GI} \in \mathrm{DTIME}[n^{\sqrt{n}}]$.)

Logics Characterizing Graphs

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

$$G \equiv_{\mathcal{L}} H \quad \Leftrightarrow \quad G \cong H \,.$$

Logics Characterizing Graphs

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

$$G \equiv_{\mathcal{L}} H \quad \Leftrightarrow \quad G \cong H \, .$$

- 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,

$$G \equiv_{\mathcal{L}} H \quad \Leftrightarrow \quad G \cong H \, .$$

- 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 $\mathrm{DTIME}[n^k \log n]$.

Logics Characterizing Graphs

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

$$G \equiv_{\mathcal{L}} H \quad \Leftrightarrow \quad G \cong H\,.$$

- 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 $\mathrm{DTIME}[n^k \log n]$.

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 $\mathcal{G}$.

Logics Characterizing Graphs

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

$$G \equiv_{\mathcal{L}} H \quad \Leftrightarrow \quad G \cong H \, .$$

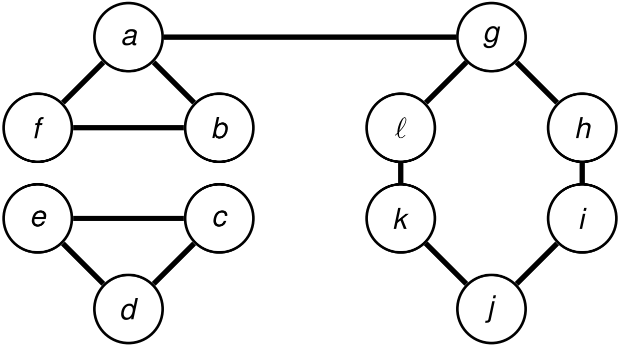
- C^2 characterizes almost all random graphs.
- C^2 characterizes all trees.
- C^3 characterizes all graphs of color class size 3.

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

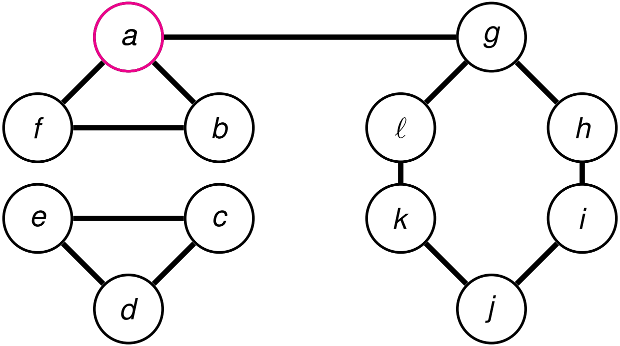
Cor. If C^k characterizes all graphs in a class of graphs $\mathcal{G}$ that is closed under particularizing, then $\mathcal{G}$ admits C^k canonization, and thus FPC captures **OIP** over $\mathcal{G}$.

proof: Apply arbitrary $\mathrm{FO(LFP)}$ formula to the canonical form of the input graph. □

Particularizing Means Uniquely Coloring Some Vertex



Particularizing Means Uniquely Coloring Some Vertex



Is FPC Equal to **OIP**?

- Is FPC Equal to **OIP**?

Is FPC Equal to **OIP**?

- Is FPC Equal to **OIP**?
- Does C^4 characterize all graphs?

Is FPC Equal to **OIP**?

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

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

Is FPC Equal to **OIP**?

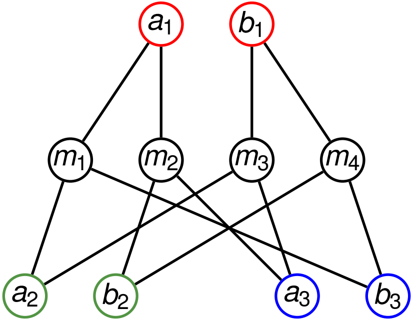
- Is FPC Equal to **OIP**?
- Does C^4 characterize all graphs?
- If yes, then $\mathrm{FPC} = $ **OIP** and for all graphs, $G \cong H \Leftrightarrow G \equiv_{C^4} H$.

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

Thm. [CFI] No!

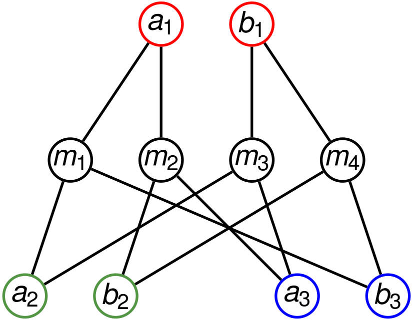
A simple graph property (now called the CFI property) checkable in $\mathrm{DTIME}[n]$, requires $v = \Omega(n)$ variables to express in C^v. Thus, CFI $\in$ **OIP** $- \mathrm{FPC}$

Proof of CFI Thm



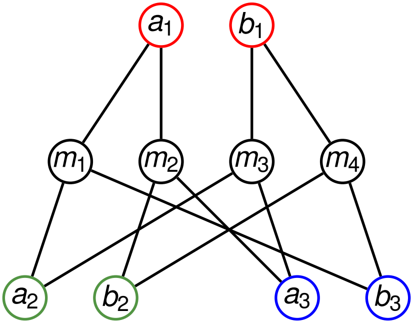
Proof of CFI Thm

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



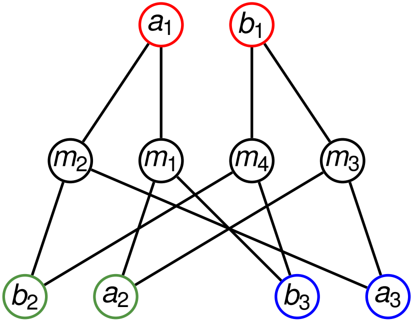
Proof of CFI Thm

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

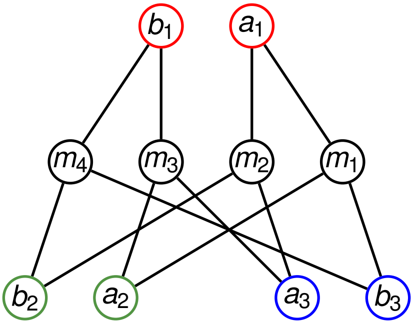
CFI Gadget X: Each m_i adjacent to an even number of a_j's.
Automorphisms of X: switch an **even number** of $(a_i b_i)$ pairs.



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: $(a_1 b_1)(a_2 b_2)(m_1 m_4)(m_2 m_3)$

G_n

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

G_n

- 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.

G_n

- 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.

G_n

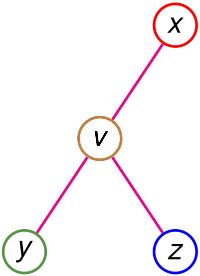
- Let G_n be a regular, degree 3 graph with $O(n)$ vertices, color class size 1 and separator size n.
- If we remove any n vertices from G_n, it still has a connected component with more than $|V^{G_n}|/2$ vertices.
- Such regular degree 3 graphs with linear-size separators exist.
- Color class size 1 means every vertex of G_n has a unique color.

G_n

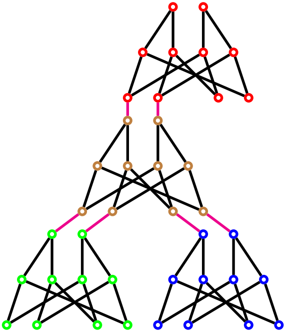
- Let G_n be a regular, degree 3 graph with $O(n)$ vertices, color class size 1 and separator size n.
- If we remove any n vertices from G_n, it still has a connected component with more than $|V^{G_n}|/2$ vertices.
- Such regular degree 3 graphs with linear-size separators exist.
- Color class size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

G_n

- 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.

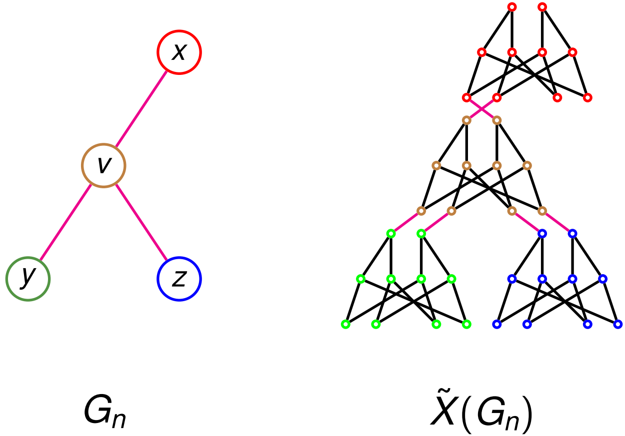


G_n

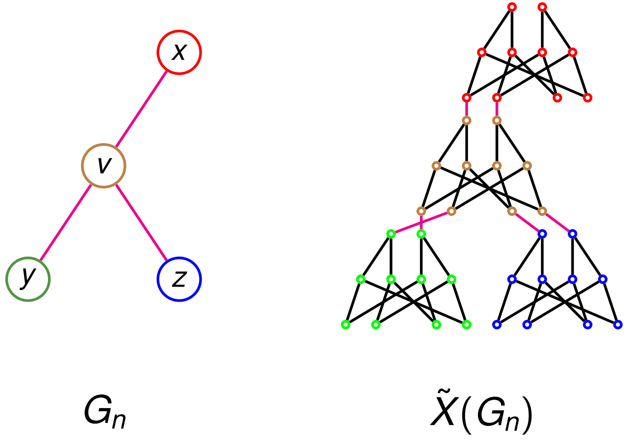


$X(G_n)$

$X(G_n)$: 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

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

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.

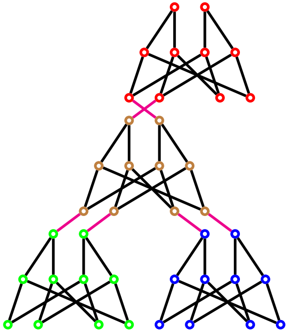
Is it $X(G_n)$ or $\tilde{X}(G_n)$?

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

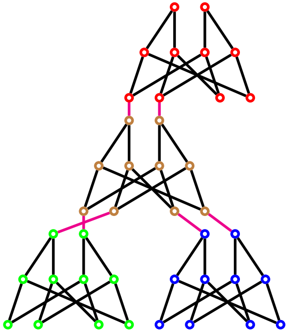
Then $X'(G_n) \cong X(G_n)$ iff m is even and

$$X'(G_n) \cong \tilde{X}(G_n) \text{ iff } m \text{ is odd.}$$

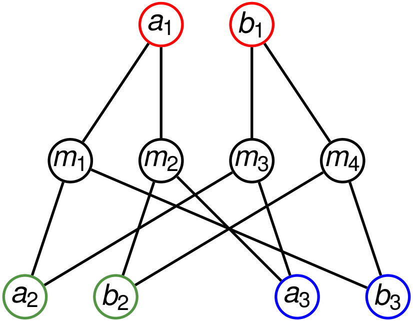
proof: Using the automorphisms of X, we can move any two flips towards each other until they eliminate each other. $\square$



$\tilde{X}(G_n)$

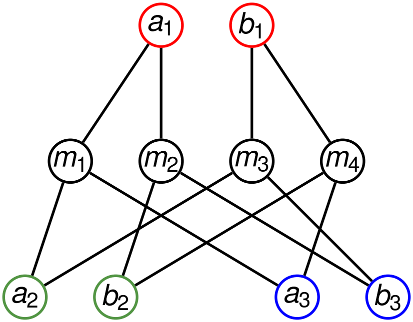


$\tilde{X}(G_n)$



X

Every one of the m_i's is connected to an even number of a_j's.



$\tilde{X}$

Every one of the m_i's is connected to an odd number of a_j's.

The CFI Problem

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

The CFI Problem

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

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

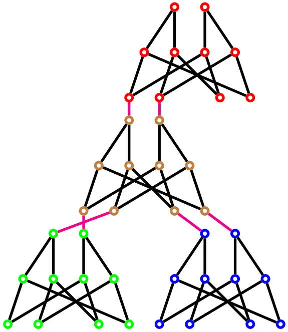
The CFI Problem

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

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

proof Use the ordering to label boundary pairs a_i, b_i when $a_i \le b_i$. Then count the number, m, of flips of vertices and edges mod 2. $X'(G) \in$ CFI iff m is even. $\square$

.



$\tilde{X}(G_n)$

Thm. CFI $\in$ **OIP** $-$ FPC.

Thm. CFI $\in$ **OIP** $- \mathrm{FPC}$.

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

Thm. CFI $\in$ **OIP** $- \mathrm{FPC}$.

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

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

Thm. CFI $\in$ **OIP** $-$ FPC.

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

Counting doesn't help since $\mathrm{cc}(X(G_n)) = 4$. 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. CFI $\in$ **OIP** $-$ FPC.

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

Counting doesn't help since $\mathrm{cc}(X(G_n)) = 4$. 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.

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.

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.

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 $w \in C_m \cap C_{m+1}$.

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. $\square$

Recap

We have shown that the linear-time CFI problem is in **OIP** $-\mathrm{FPC}$.

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

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 $\mathcal{G}$ of graphs that excludes some minor is characterized by C^k for some fixed k.

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

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 $\mathcal{G}$ 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.

Going Beyond FPC

Two other languages are candidates for capturing **OIP**:

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.

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.

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 expresible in CPT and in Rank Logic, thus these are strict extenstions of FPC.

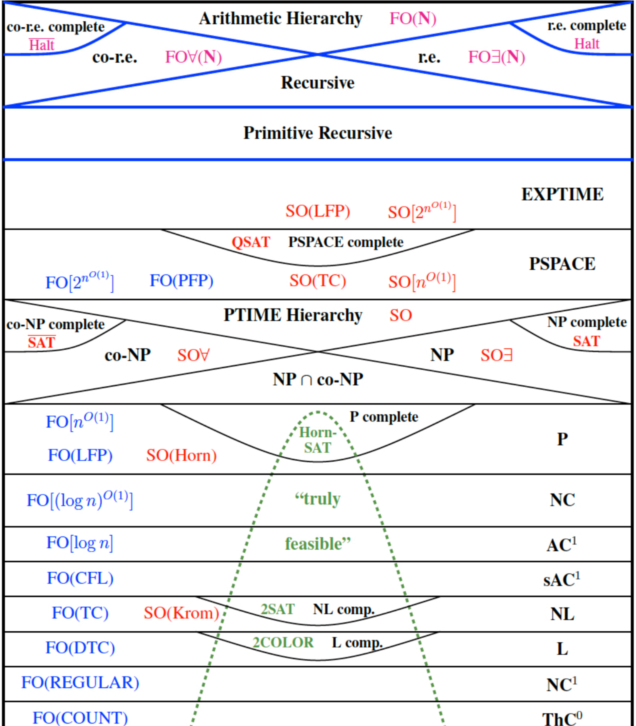
Going Beyond FPC

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 expresible in CPT and in Rank Logic, thus these are strict extenstions 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)
co-r.e. complete
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
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