A Fine-Grained Analogue of Schaefer's Theorem in P: Dichotomy of $\exists^k\forall$-Quantified First-Order Graph Properties

Karl Bringmann[1]

Nick Fischer[1]

Marvin Künnemann[1]

[1] Max Planck Institute for Informatics, Saarland Informatics Campus (SIC), Saarbrücken

max planck institut informatik

SIC Saarland Informatics Campus

First-Order Property Model-Checking

Fix a first-order property ψ. The model-checking problem for ψ asks to check whether ψ is true on a given structure (e.g. graph).

First-Order Property Model-Checking

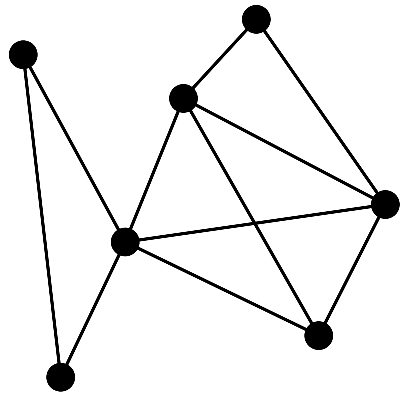
Fix a first-order property ψ. The model-checking problem for ψ asks to check whether ψ is true on a given structure (e.g. graph).

$$\exists x\, \exists y\, \exists z\colon \; \overline{E(x,y)} \wedge \overline{E(y,z)} \wedge \overline{E(z,x)}$$

First-Order Property Model-Checking

Fix a first-order property ψ. The model-checking problem for ψ asks to check whether ψ is true on a given structure (e.g. graph).

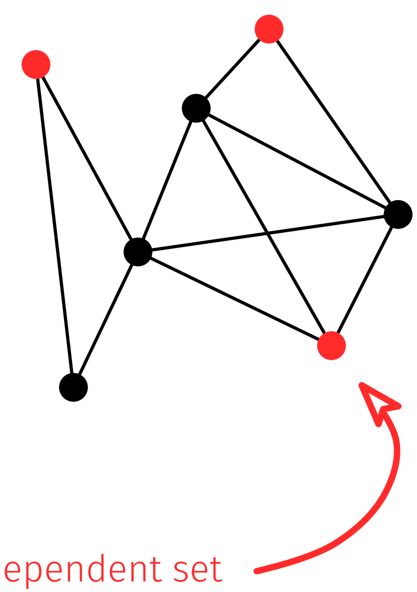
$$\exists x\, \exists y\, \exists z: \overline{E(x, y)} \wedge \overline{E(y, z)} \wedge \overline{E(z, x)}$$



First-Order Property Model-Checking

Fix a first-order property ψ. The model-checking problem for ψ asks to check whether ψ is true on a given structure (e.g. graph).

$$\exists x\, \exists y\, \exists z: \overline{E(x,y)} \wedge \overline{E(y,z)} \wedge \overline{E(z,x)}$$

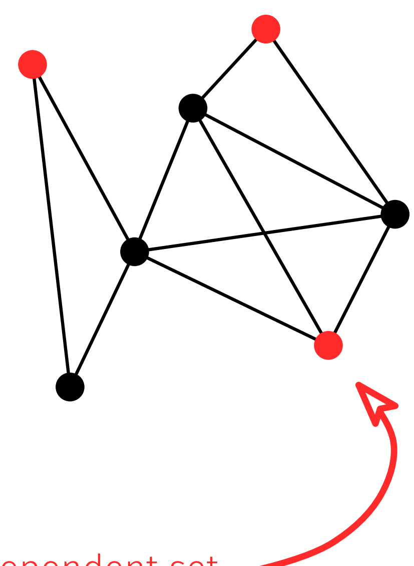


3-independent set

First-Order Property Model-Checking

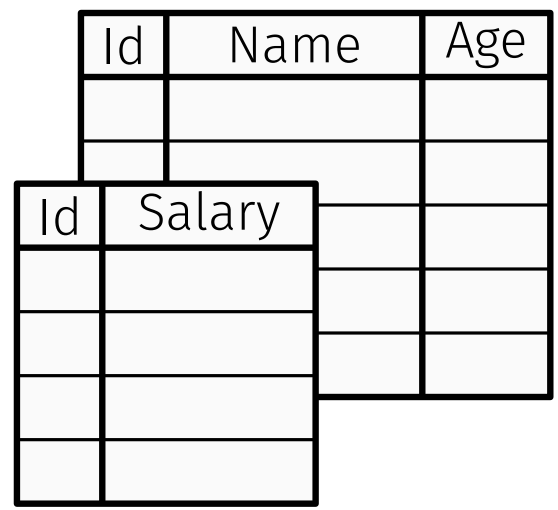
Fix a first-order property ψ. The model-checking problem for ψ asks to check whether ψ is true on a given structure (e.g. graph).

$$\exists x\, \exists y\, \exists z: \overline{E(x,y)} \wedge \overline{E(y,z)} \wedge \overline{E(z,x)}$$



3-independent set

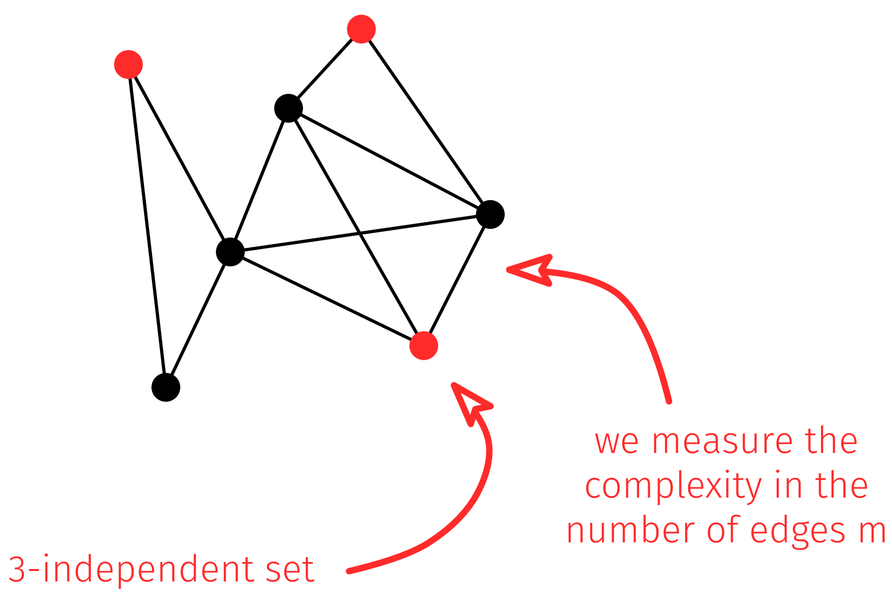
SQL



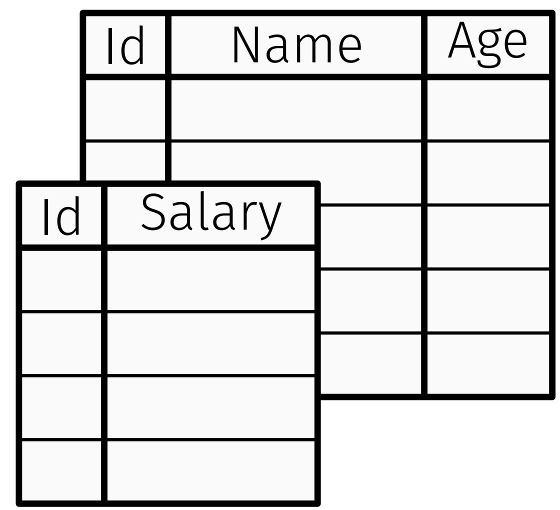
First-Order Property Model-Checking

Fix a first-order property ψ. The model-checking problem for ψ asks to check whether ψ is true on a given structure (e.g. graph).

$$\exists x\, \exists y\, \exists z\colon\ \overline{E(x,y)} \land \overline{E(y,z)} \land \overline{E(z,x)}$$



SQL

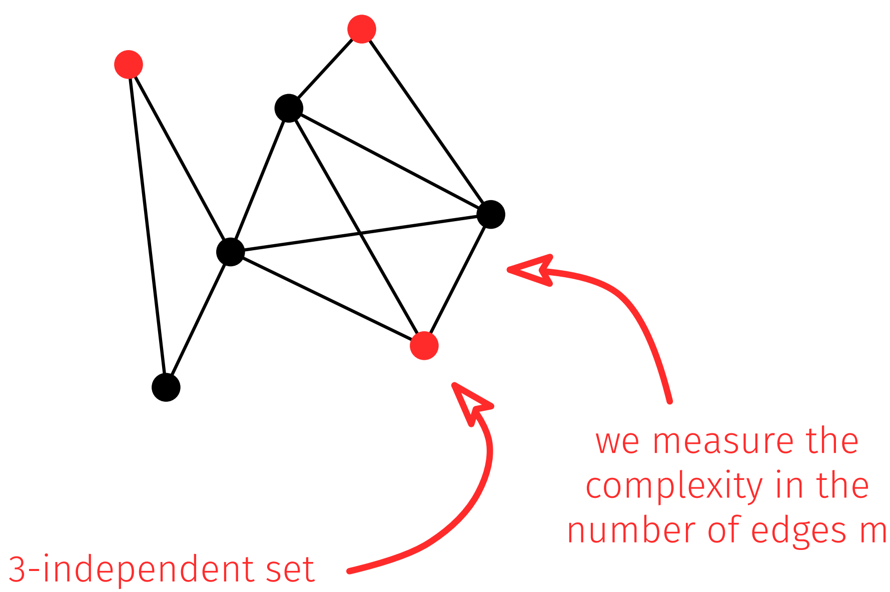


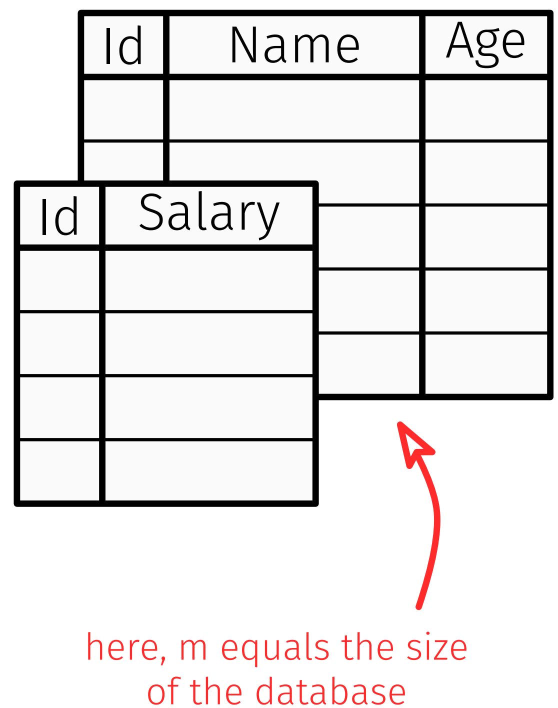
First-Order Property Model-Checking

Fix a first-order property ψ. The model-checking problem for ψ asks to check whether ψ is true on a given structure (e.g. graph).

$$\exists x\ \exists y\ \exists z: \overline{E(x,y)} \wedge \overline{E(y,z)} \wedge \overline{E(z,x)}$$

SQL





Orthogonal Vectors

Given two sets $X_1, X_2 \subseteq \{0,1\}^d$ of size n, check whether there exists an orthogonal pair $x_1 \in X_1, x_2 \in X_2$

1	1	0
0	1	0
1	1	1
1	0	1

orth.

0	1	1
1	1	0
1	0	0
1	1	1

Orthogonal Vectors

Given two sets $X_1, X_2 \subseteq \{0,1\}^d$ of size n, check whether there exists an orthogonal pair $x_1 \in X_1, x_2 \in X_2$

1	1	0
0	1	0
1	1	1
1	0	1

orth.

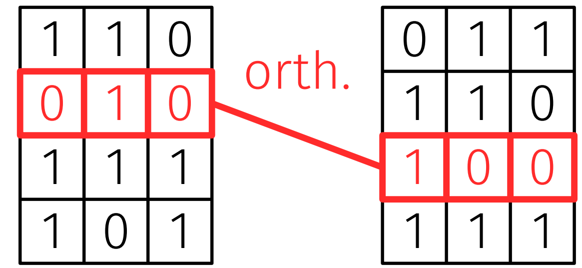
0	1	1
1	1	0
1	0	0
1	1	1

requires time
$n^{2-o(1)} \cdot \mathrm{poly}(d)$
under SETH

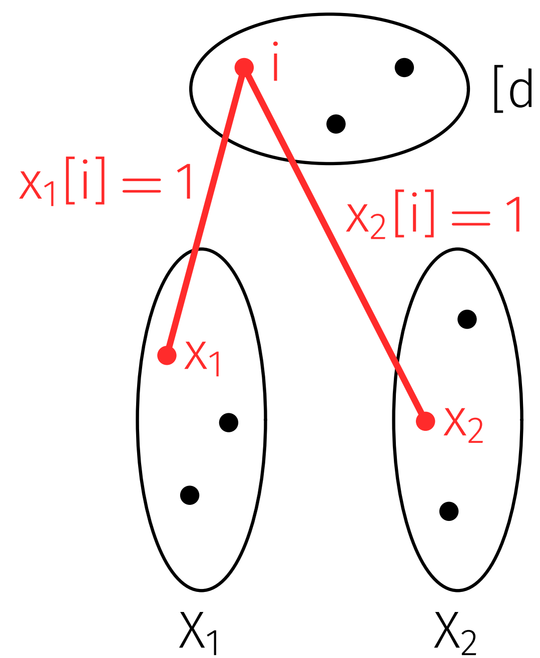
Orthogonal Vectors

Given two sets $X_1, X_2 \subseteq \{0,1\}^d$ of size n, check whether there exists an orthogonal pair $x_1 \in X_1, x_2 \in X_2$

requires time
$n^{2-o(1)} \cdot \mathrm{poly}(d)$
under SETH

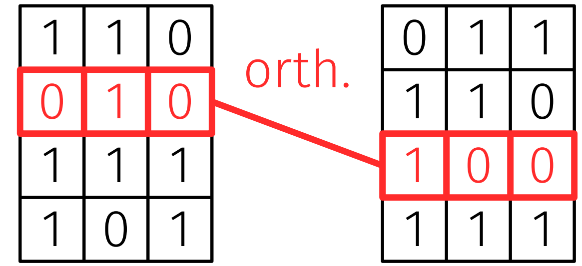


$$\exists x_1 \in X_1 \, \exists x_2 \in X_2 \, \forall i \in [d]: \\ x_1[i] = 0 \lor x_2[i] = 0$$



Orthogonal Vectors

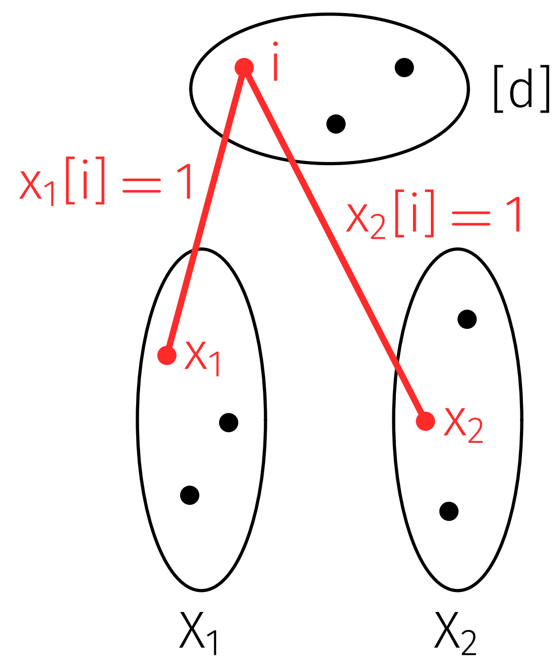
Given two sets $X_1, X_2 \subseteq \{0,1\}^d$ of size n, check whether there exists an orthogonal pair $x_1 \in X_1, x_2 \in X_2$



requires time
$n^{2-o(1)} \cdot \mathrm{poly}(d)$
under SETH

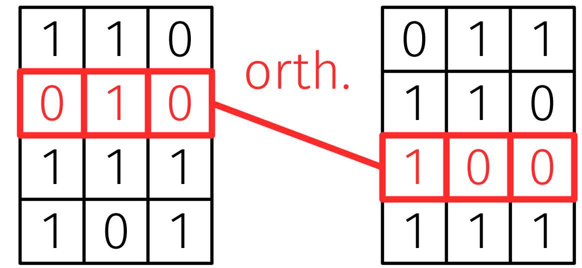
here, m equals the
total number of
1-entries

$$\exists x_1 \in X_1 \, \exists x_2 \in X_2 \, \forall i \in [d]: \\ x_1[i] = 0 \lor x_2[i] = 0$$



k-Orthogonal Vectors

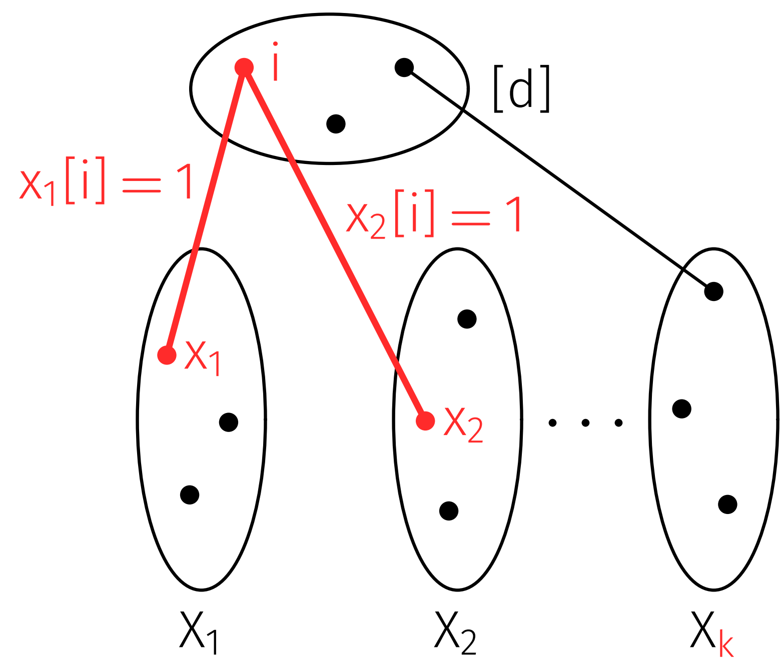
Given two sets $X_1, X_2 \subseteq \{0,1\}^d$ of size n, check whether there exists an orthogonal pair $x_1 \in X_1, x_2 \in X_2$



requires time
$n^{2-o(1)} \cdot \text{poly}(d)$
under SETH

here, m equals the
total number of
1-entries

$$\exists x_1 \in X_1 \;\ldots\; \exists x_k \in X_k \; \forall i \in [d]:$$
$$x_1[i] = 0 \vee \ldots \vee x_k[i] = 0$$



Our Starting Point

- Each $(k+1)$-quantifier first-order query can be checked in time $O(m^k)$

Our Starting Point

- Each $(k+1)$-quantifier first-order query can be checked in time $O(m^k)$
- (Sparse) k-OV is complete for the class of $(k+1)$-quantifier properties [Gao, Impagliazzo, Kolokolova, Williams '17]

Our Starting Point

- Each $(k+1)$-quantifier first-order query can be checked in time $O(m^k)$
- (Sparse) k-OV is complete for the class of $(k+1)$-quantifier properties [Gao, Impagliazzo, Kolokolova, Williams '17]
- All complete properties require time $m^{k-o(1)}$ under SETH

Our Starting Point

- Each $(k+1)$-quantifier first-order query can be checked in time $O(m^k)$
- (Sparse) k-OV is complete for the class of $(k+1)$-quantifier properties [Gao, Impagliazzo, Kolokolova, Williams '17]
- All complete properties require time $m^{k-o(1)}$ under SETH

What about the others?
Can we classify queries according to their complexity?

Our Starting Point

- Each $(k+1)$-quantifier first-order query can be checked in time $O(m^k)$
- (Sparse) k-OV is complete for the class of $(k+1)$-quantifier properties [Gao, Impagliazzo, Kolokolova, Williams '17]
- All complete properties require time $m^{k-o(1)}$ under SETH

What about the others?
Can we classify queries according to their complexity?

$$O(m^k) \ \ \textit{vs.} \ \ O(m^{k-0.01})$$

Our Starting Point

- Each $(k+1)$-quantifier first-order query can be checked in time $O(m^k)$
- (Sparse) k-OV is complete for the class of $(k+1)$-quantifier properties [Gao, Impagliazzo, Kolokolova, Williams '17]
- All complete properties require time $m^{k-o(1)}$ under SETH

Constraint satisfaction problems	First-order properties
3-SAT is NP-complete [Cook '71]	k-OV is FOP-complete [GIKW '17]
Every Boolean CSP is either in P or NP-complete [Schaefer '78]	?

Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

Our Main Result: A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{array}{c} \exists x_1 \ \ldots \ \exists x_k \ \forall y: \\ \varphi(E(x_1, y), \ldots, E(x_k, y)) \end{array}$$

Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{array}{c}\exists x_1 \ \dots \ \exists x_k \ \forall y: \\ \varphi(E(x_1, y), \dots, E(x_k, y))\end{array}$$

Boolean function
$\varphi : \{0,1\}^k \to \{0,1\}$

Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{array}{c} \exists x_1 \ \dots \ \exists x_k \ \forall y: \\ \varphi(E(x_1, y), \dots, E(x_k, y)) \end{array}$$

Boolean function
$\varphi : \{0, 1\}^k \to \{0, 1\}$

The hardness of ψ is the largest number $h \in \{0, \dots, k\}$, such that

"ψ is h-OV-like"

Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{array}{c} \exists x_1 \; \dots \; \exists x_k \; \forall y: \\ \varphi(E(x_1, y), \dots, E(x_k, y)) \end{array}$$

Boolean function
$\varphi : \{0,1\}^k \to \{0,1\}$

The hardness of ψ is the largest number $h \in \{0, \dots, k\}$, such that

for any subset $J \subseteq [k]$ of $k - h$ inputs,

there exists an assignment $\alpha : J \to \{0,1\}$,

so that $\varphi|_\alpha$ has exactly one falsifying assignment

Our Main Result: A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{array}{c}\exists x_1 \; \dots \; \exists x_k \; \forall y: \\ \varphi(E(x_1, y), \dots, E(x_k, y))\end{array}$$

Boolean function
$\varphi : \{0,1\}^k \to \{0,1\}$

The hardness of ψ is the largest number $h \in \{0, \dots, k\}$, such that

"ψ is h-OV-like"

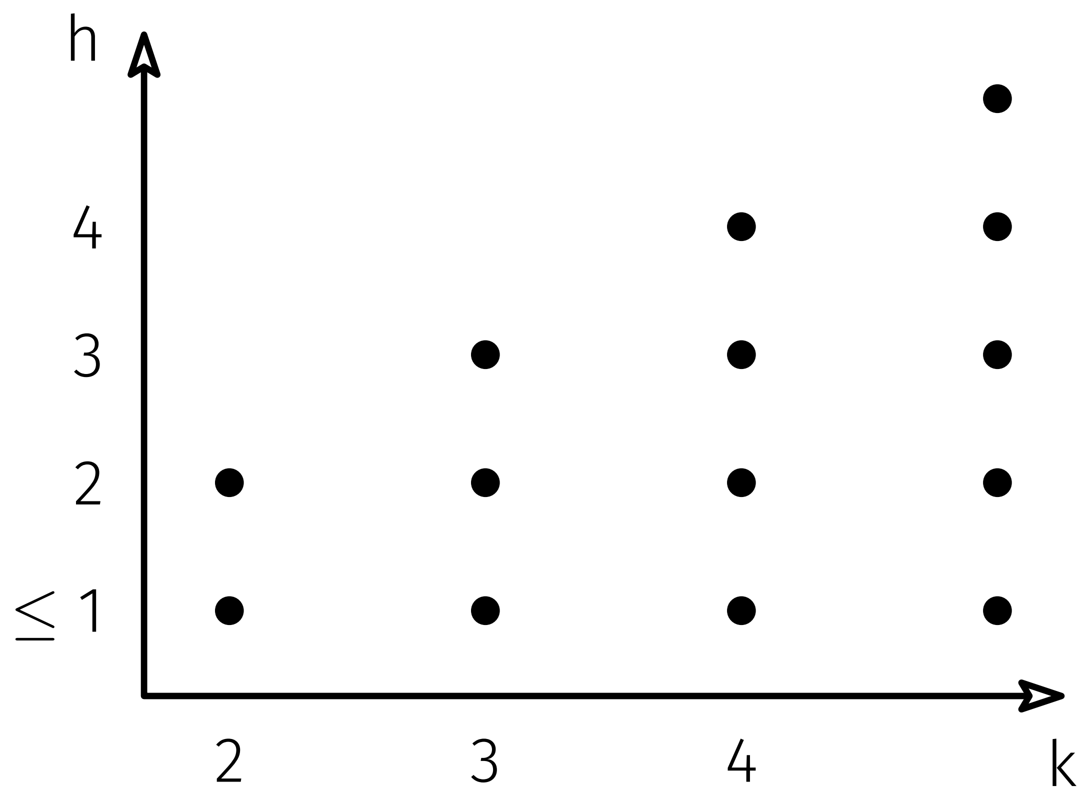
Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{matrix} \exists x_1 \ \dots \ \exists x_k \ \forall y: \\ \varphi(E(x_1, y), \dots, E(x_k, y)) \end{matrix}$$

Boolean function
$\varphi : \{0,1\}^k \to \{0,1\}$

The hardness of ψ is the largest number $h \in \{0, \dots, k\}$, such that

"ψ is h-OV-like"



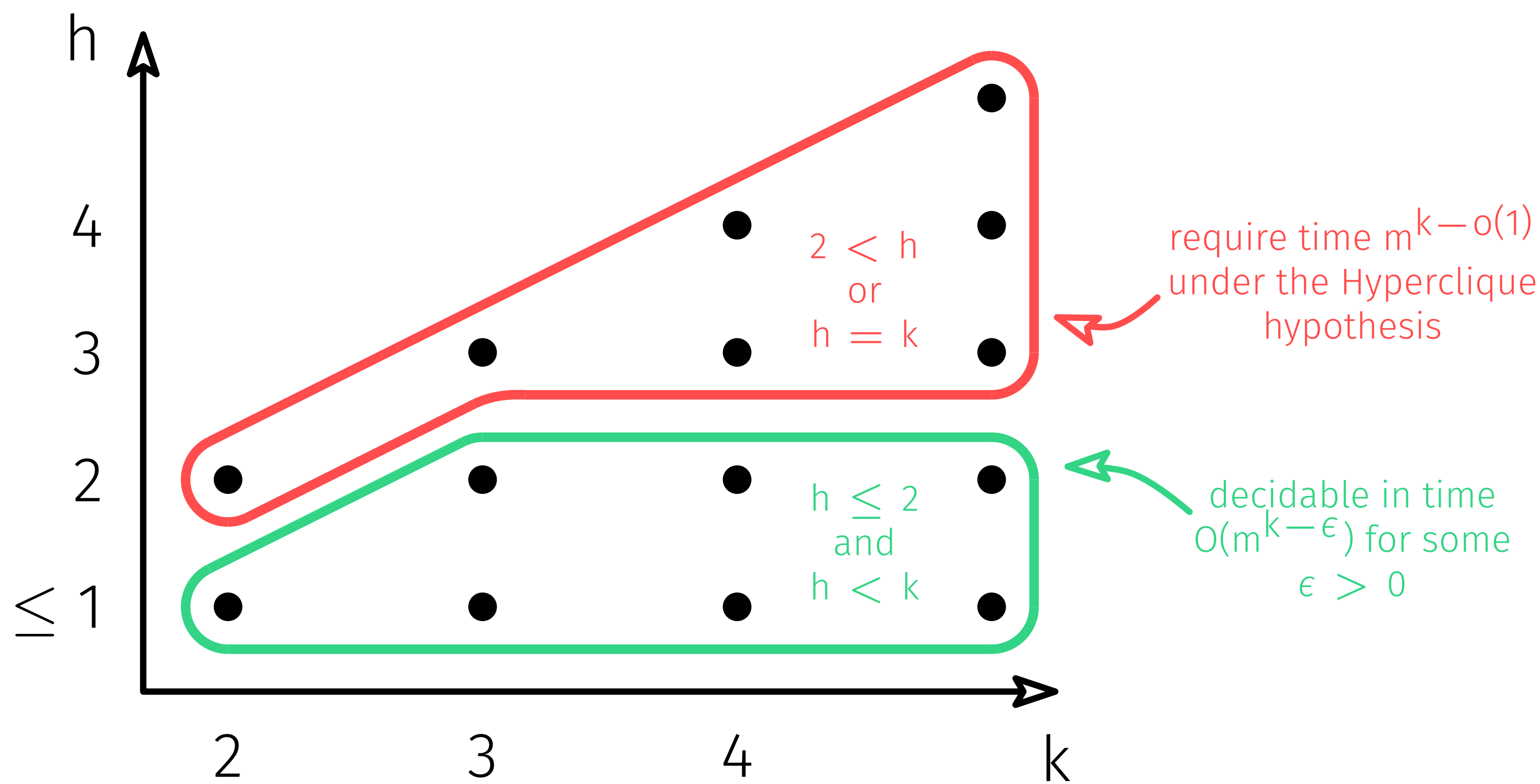
Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{array}{c} \exists x_1 \ \ldots \ \exists x_k \ \forall y: \\ \varphi(E(x_1, y), \ldots, E(x_k, y)) \end{array}$$

Boolean function
$\varphi : \{0,1\}^k \to \{0,1\}$

The hardness of ψ is the largest number $h \in \{0, \ldots, k\}$, such that

"ψ is h-OV-like"



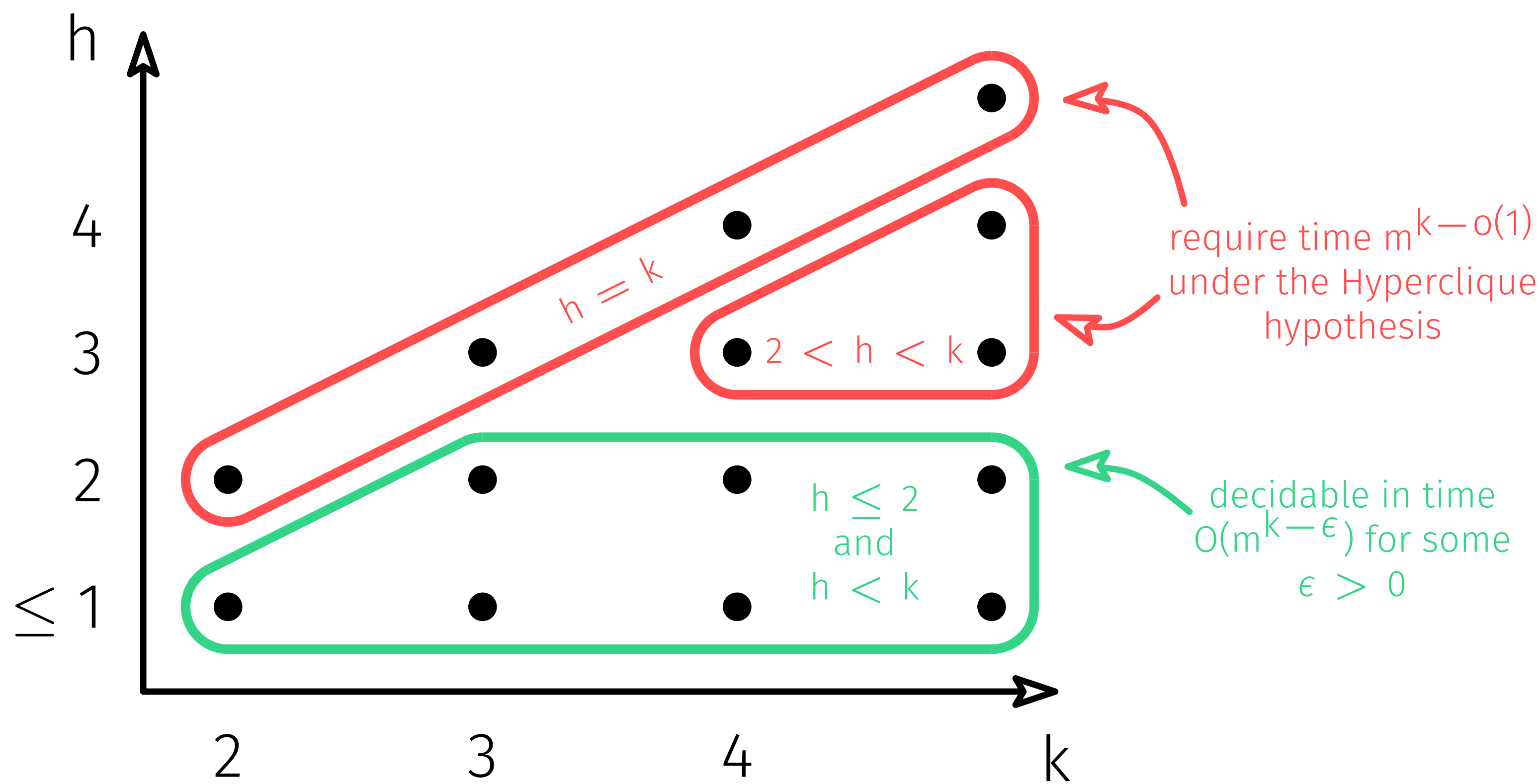
Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{array}{c}\exists x_1 \ \dots \ \exists x_k \ \forall y: \\ \varphi(E(x_1, y), \dots, E(x_k, y))\end{array}$$

Boolean function
$\varphi : \{0,1\}^k \to \{0,1\}$

The hardness of ψ is the largest number $h \in \{0, \dots, k\}$, such that

"ψ is h-OV-like"



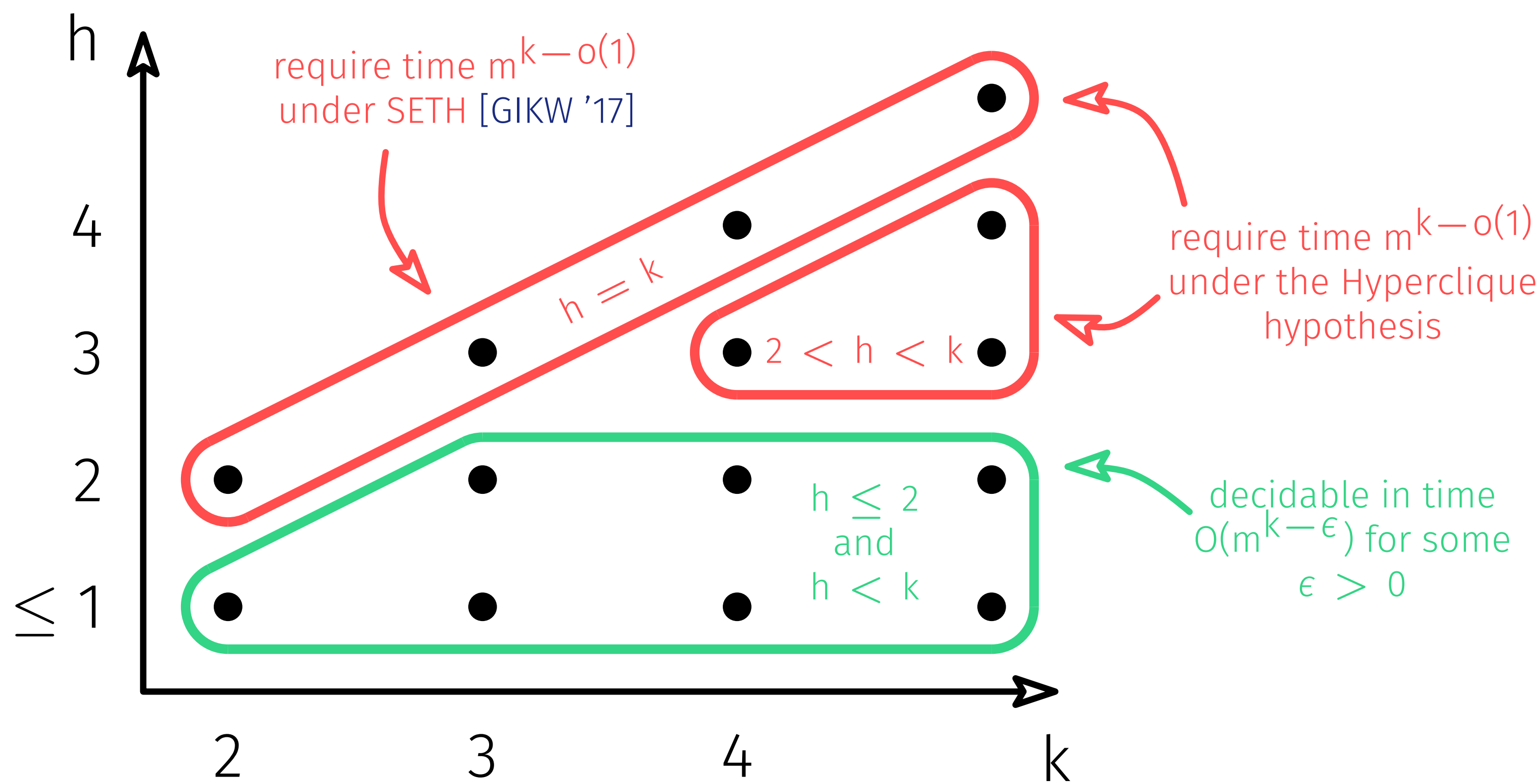
Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{array}{c}\exists x_1 \ldots \exists x_k \forall y: \\ \varphi(E(x_1, y), \ldots, E(x_k, y))\end{array}$$

Boolean function
$\varphi : \{0,1\}^k \to \{0,1\}$

The hardness of ψ is the largest number $h \in \{0, \ldots, k\}$, such that

"ψ is h-OV-like"



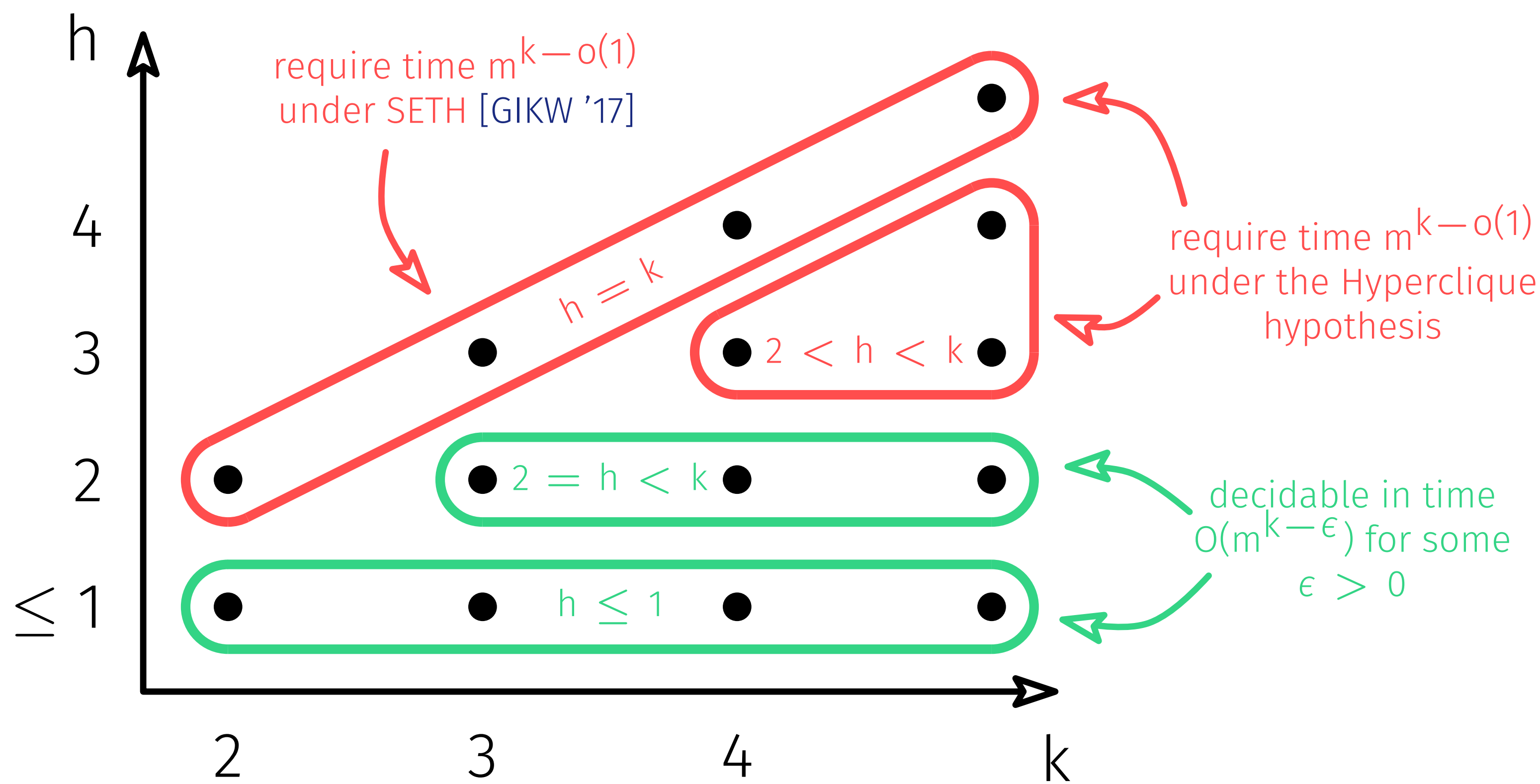
Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{array}{c}\exists x_1 \; \dots \; \exists x_k \; \forall y: \\ \varphi(E(x_1, y), \dots, E(x_k, y))\end{array}$$

Boolean function
$\varphi : \{0,1\}^k \to \{0,1\}$

The hardness of ψ is the largest number $h \in \{0, \dots, k\}$, such that

"ψ is h-OV-like"



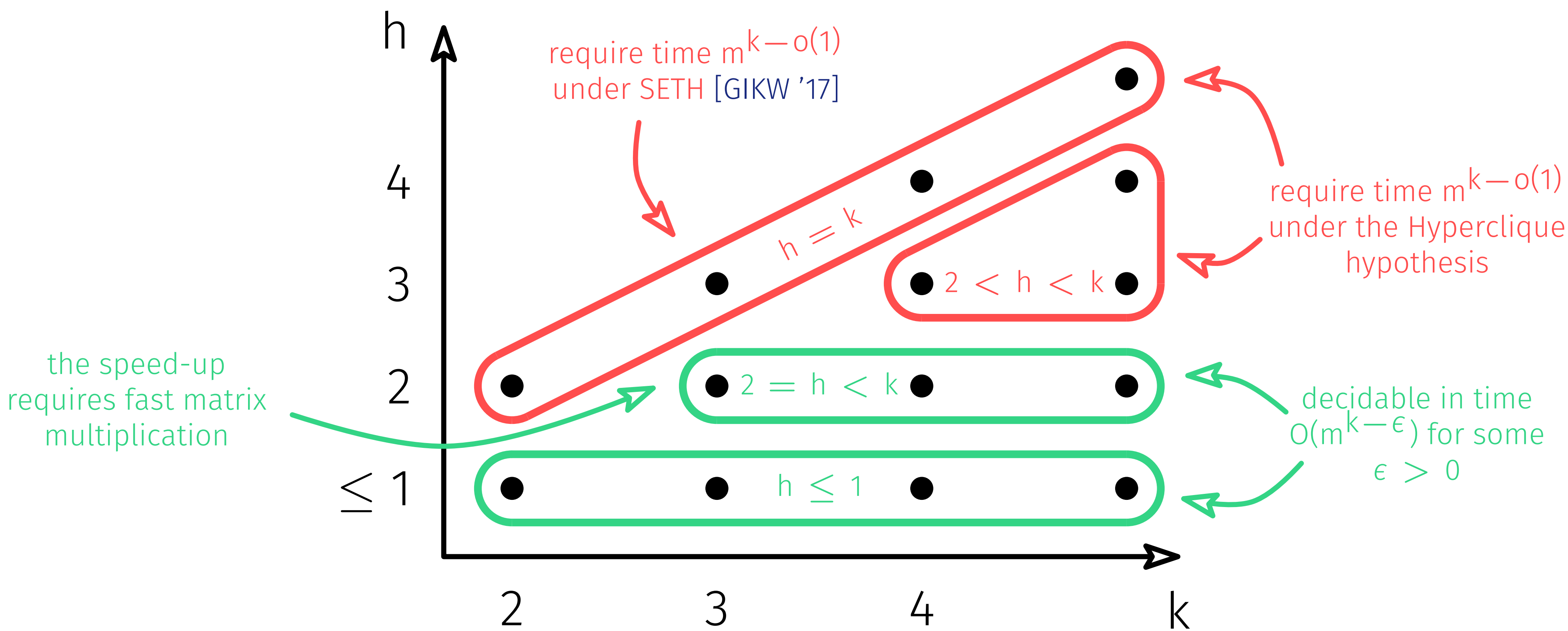
Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \begin{array}{c}\exists x_1 \ \ldots \ \exists x_k \ \forall y: \\ \varphi(E(x_1, y), \ldots, E(x_k, y))\end{array}$$

Boolean function
$\varphi : \{0,1\}^k \to \{0,1\}$

The hardness of ψ is the largest number $h \in \{0, \ldots, k\}$, such that

"ψ is h-OV-like"



Lower Bounds for Properties of Hardness $h \geq 3$

Hypothesis: h-uniform Hyperclique

For $h \geq 3$, detecting a k-clique in an h-hypergraph requires time $n^{k-o(1)}$

Lower Bounds for Properties of Hardness $h \geq 3$

Hypothesis: h-uniform Hyperclique

For $h \geq 3$, detecting a k-clique in an h-hypergraph requires time $n^{k-o(1)}$

fails for $h \leq 2$:
$O(n^{\omega k/3})$ using fast matrix multiplication

Lower Bounds for Properties of Hardness $h \geq 3$

Hypothesis: h-uniform Hyperclique

For $h \geq 3$, detecting a k-clique in an h-hypergraph requires time $n^{k-o(1)}$

fails for $h \leq 2$: $O(n^{\omega k/3})$ using fast matrix multiplication

is implied by the assumption that MAX-3-SAT cannot be solved in time $O(2^{(1-\varepsilon)n})$ [Williams '07]

Lower Bounds for Properties of Hardness $h \geq 3$

Hypothesis: h-uniform Hyperclique

For $h \geq 3$, detecting a k-clique in an h-hypergraph requires time $n^{k-o(1)}$

fails for $h \leq 2$:
$O(n^{\omega k/3})$ using fast matrix multiplication

Strassen-like techniques are ruled out
[Lincoln, V-Williams, Williams '18]

is implied by the assumption that MAX-3-SAT cannot be solved in time $O(2^{(1-\varepsilon)n})$
[Williams '07]

Lower Bounds for Properties of Hardness $h \geq 3$

Hypothesis: h-uniform Hyperclique

For $h \geq 3$, detecting a k-clique in an h-hypergraph requires time $n^{k-o(1)}$

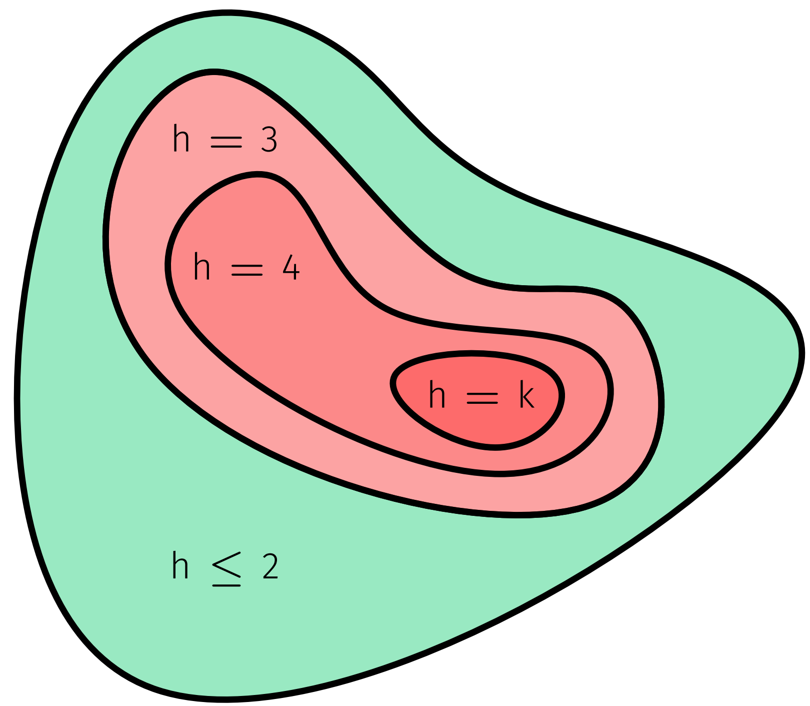
fails for $h \leq 2$: $O(n^{\omega k/3})$ using fast matrix multiplication

Strassen-like techniques are ruled out [Lincoln, V-Williams, Williams '18]

is implied by the assumption that MAX-3-SAT cannot be solved in time $O(2^{(1-\varepsilon)n})$ [Williams '07]

Our results: Hardness levels

Unless the h-uniform Hyperclique hypothesis fails, model-checking any property of hardness h requires time $m^{k-o(1)}$



Lower Bounds for Properties of Hardness $h \geq 3$

Hypothesis: h-uniform Hyperclique

For $h \geq 3$, detecting a k-clique in an h-hypergraph requires time $n^{k-o(1)}$

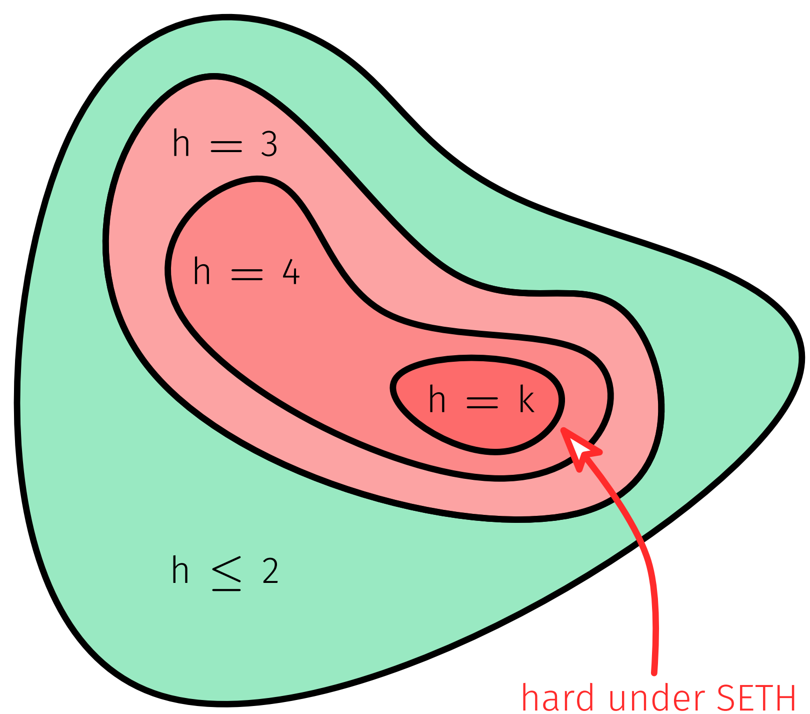
fails for $h \leq 2$: $O(n^{\omega k/3})$ using fast matrix multiplication

Strassen-like techniques are ruled out [Lincoln, V-Williams, Williams '18]

is implied by the assumption that MAX-3-SAT cannot be solved in time $O(2^{(1-\varepsilon)n})$ [Williams '07]

Our results: Hardness levels

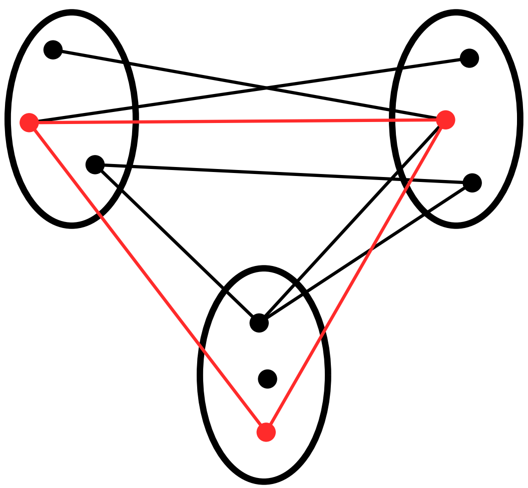
Unless the h-uniform Hyperclique hypothesis fails, model-checking any property of hardness h requires time $m^{k-o(1)}$



Build your own SUBcubic problem!

Step 1: Take the basis problem

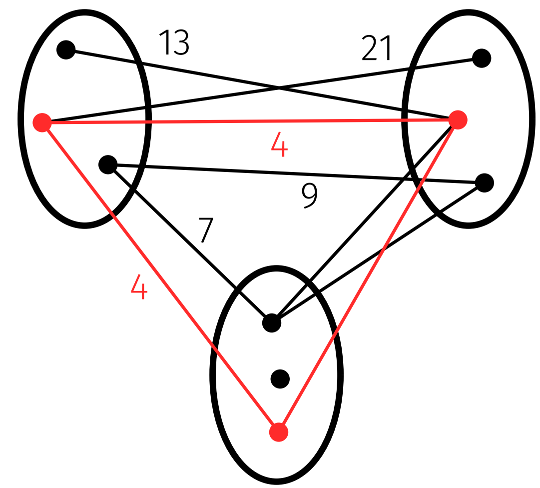
$\cancel{O(n^3)}$ $O(n^{\omega})$



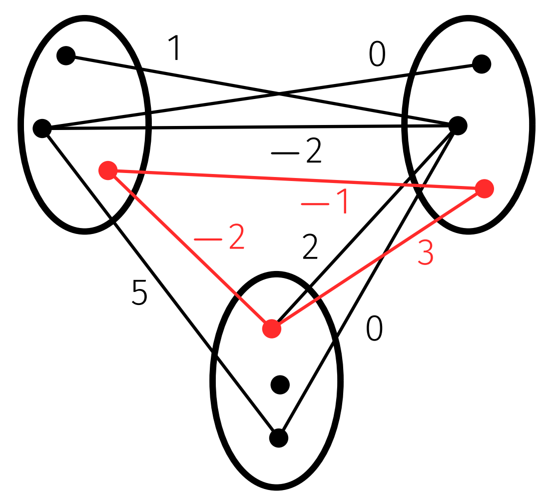
(Triangle Detection)

Step 2: Choose your toppings

$\cancel{O(n^3)}$ $O(n^{3-\varepsilon})$



(Equal Constraint)

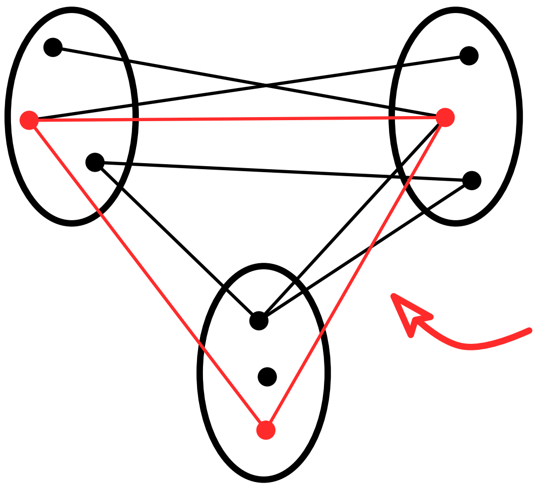


(Sum Constraint)

Build your own SUBcubic problem!

Step 1: Take the basis problem

$\cancel{\Theta(n^3)}$ $O(n^{\omega})$

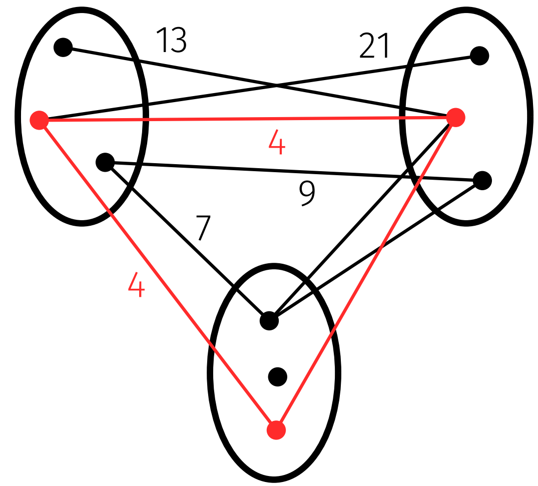


we assume tripartite graphs

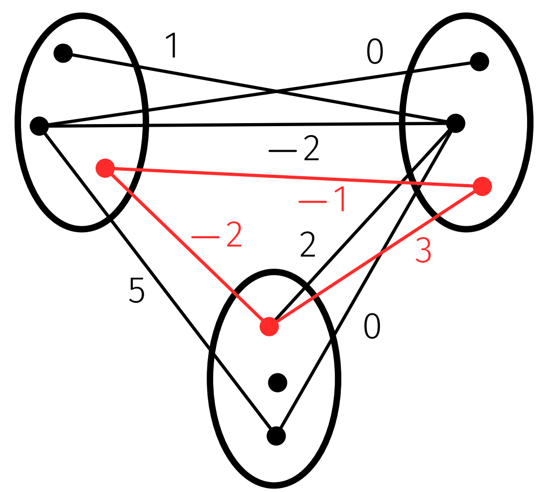
(Triangle Detection)

Step 2: Choose your toppings

$\cancel{\Theta(n^3)}$ $O(n^{3-\varepsilon})$



(Equal Constraint)

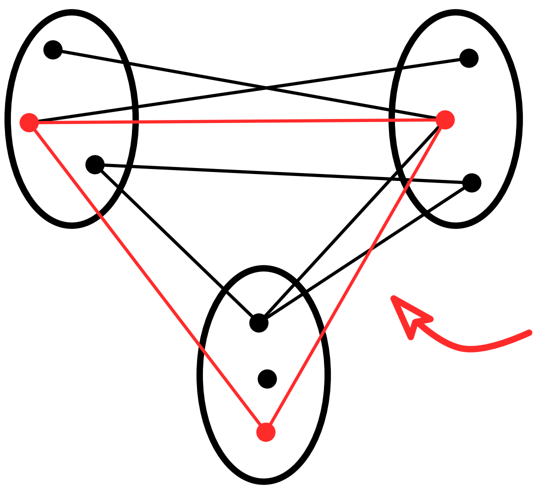


(Sum Constraint)

Build your own SUBcubic problem!

Step 1: Take the basis problem

$\cancel{\Theta(n^3)}$ $O(n^\omega)$



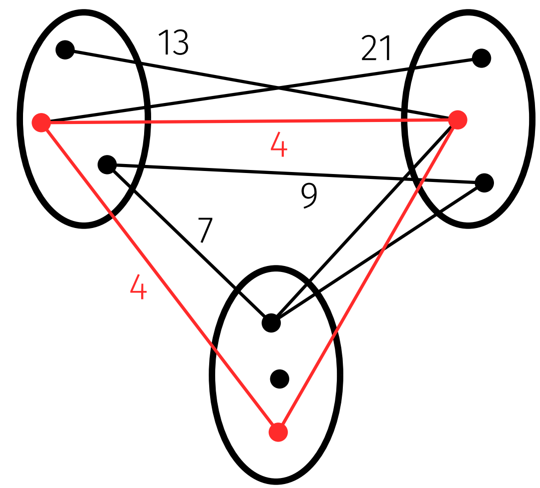
(Triangle Detection)

we assume tripartite graphs

+

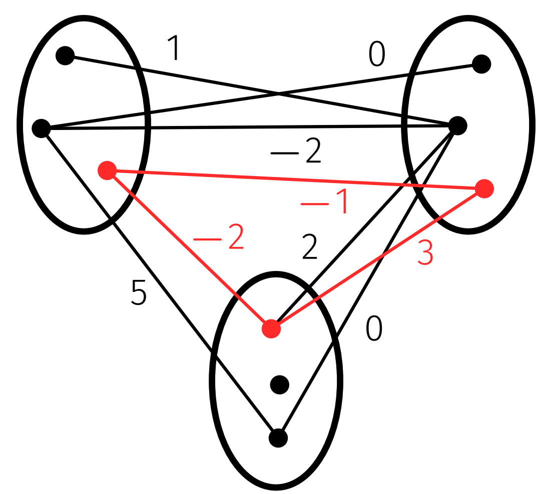
Step 2: Choose your toppings

$\cancel{\Theta(n^3)}$ $O(n^{3-\varepsilon})$



(Equal Constraint)

works for any target t
(here $t = 0$)

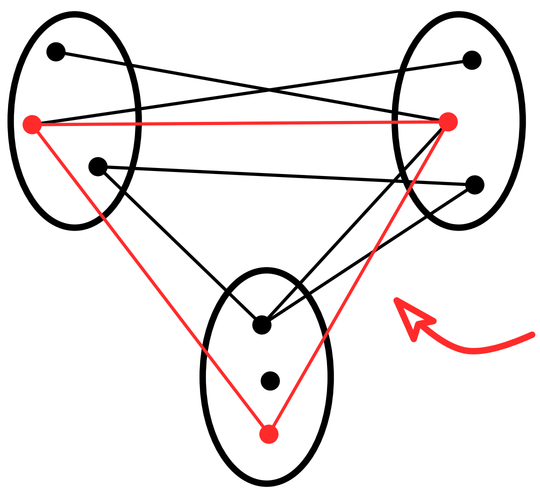


(Sum Constraint)

Build your own SUBcubic problem!

Step 1: Take the basis problem

$\cancel{\Theta(n^3)}$ $O(n^{\omega})$



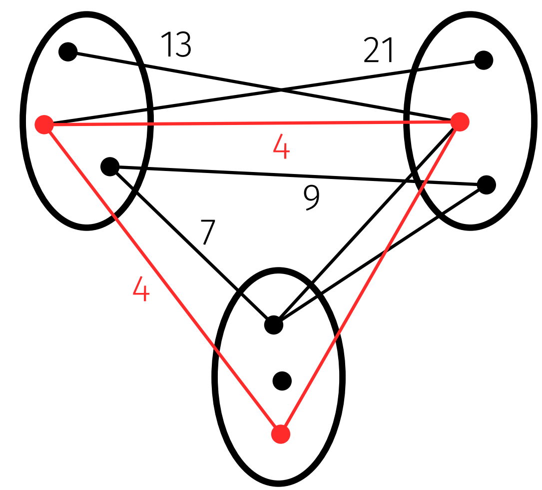
we assume tripartite graphs

(Triangle Detection)

+

Step 2: Choose your toppings

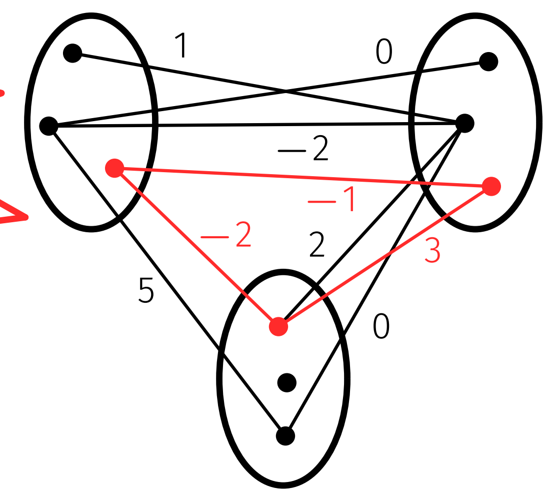
$\cancel{\Theta(n^3)}$ $O(n^{3-\varepsilon})$



(Equal Constraint)

works for any target t (here $t = 0$)

small total weight: $\sum_e |w(e)| \leq O(n^2)$

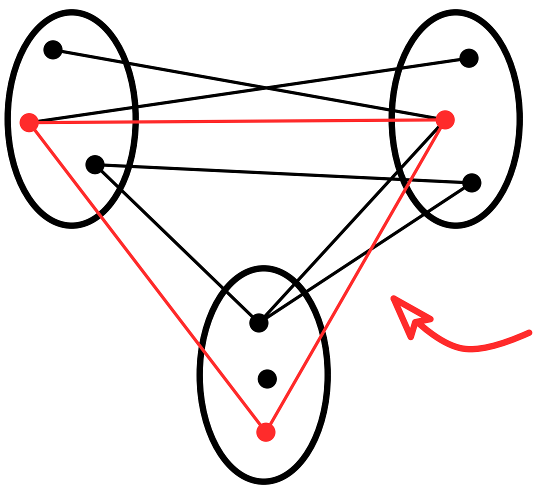


(Sum Constraint)

Build your own SUBcubic problem!

Step 1: Take the basis problem

$\cancel{O(n^3)}$ $O(n^{\omega})$

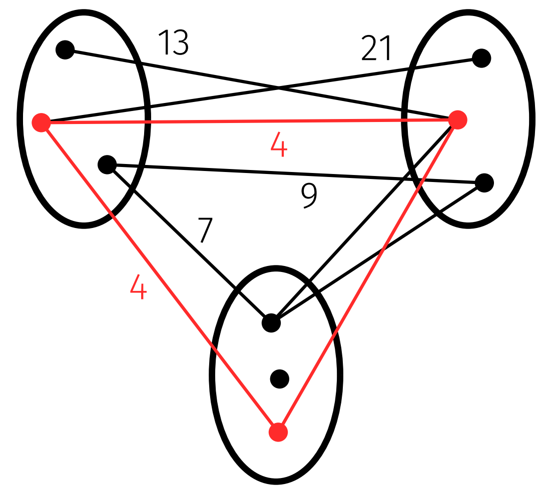


we assume tripartite graphs

(Triangle Detection)

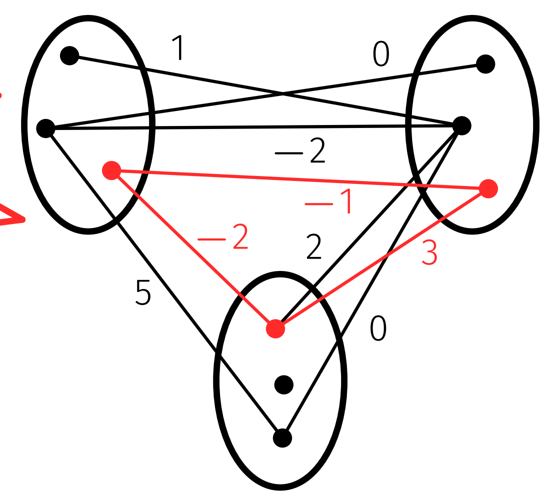
Step 2: Choose your toppings

$\cancel{O(n^3)}$ $O(n^{3-\varepsilon})$



(Equal Constraint)

works for any target t (here $t = 0$)

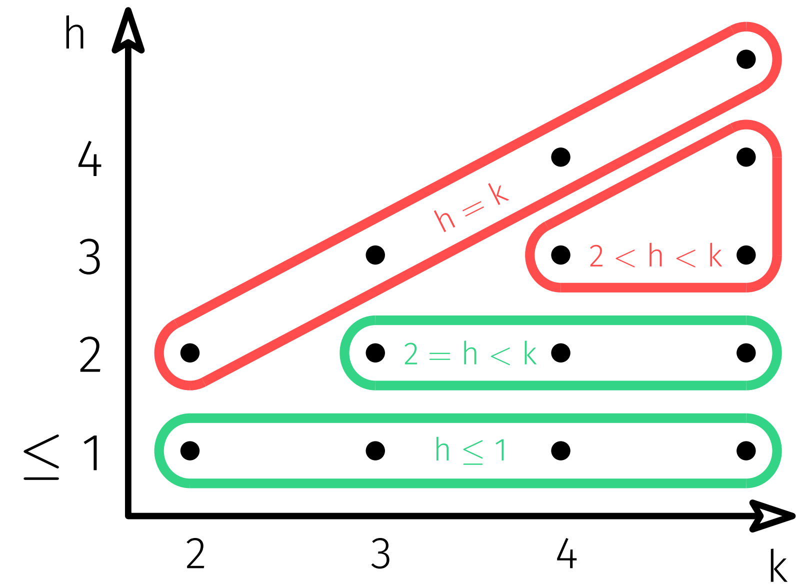


small total weight: $\sum_e |w(e)| \leq O(n^2)$

(Sum Constraint)

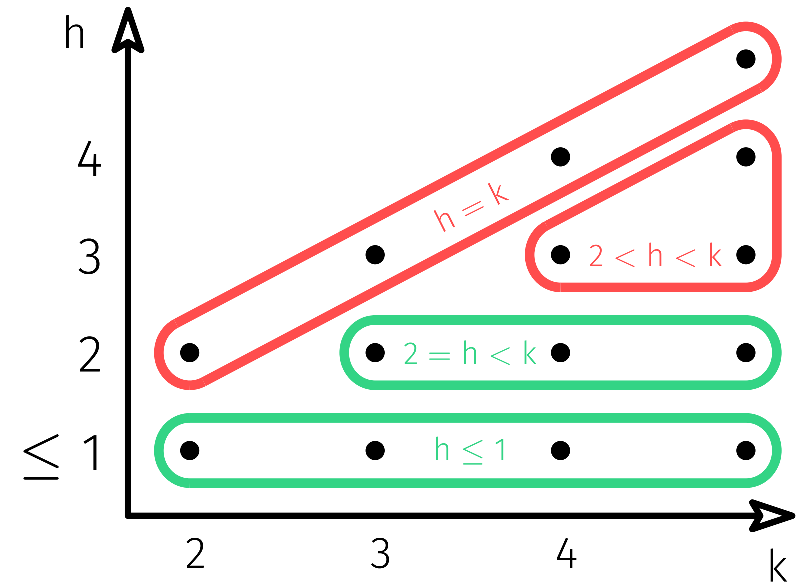
“Constrained Triangle Detection”

Algorithms for Properties of Hardness $h \leq 2$



Algorithms for Properties of Hardness $h \leq 2$

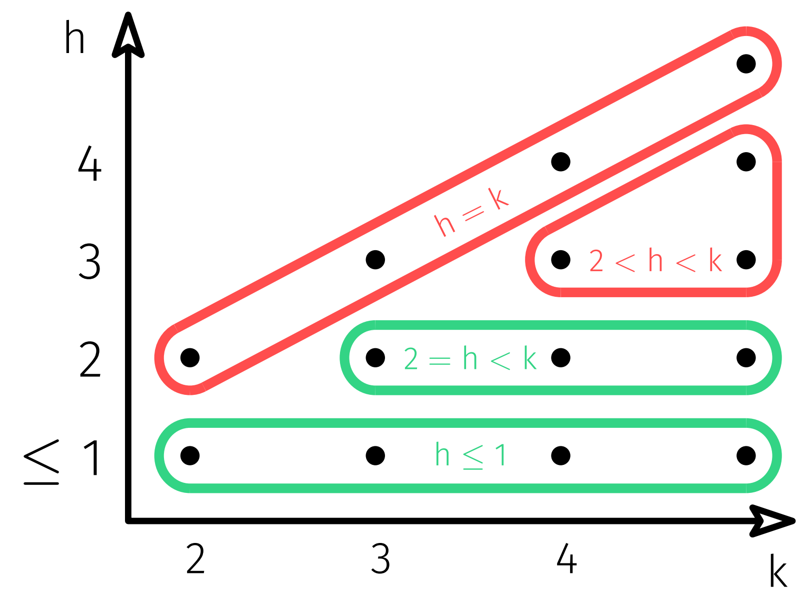
$k = 3$: Reduction to Constrained Triangles



Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

$$\exists x_1 \exists x_2 \exists x_3 \forall y:$$
$$\varphi(E(x_1, y), E(x_2, y), E(x_3, y))$$

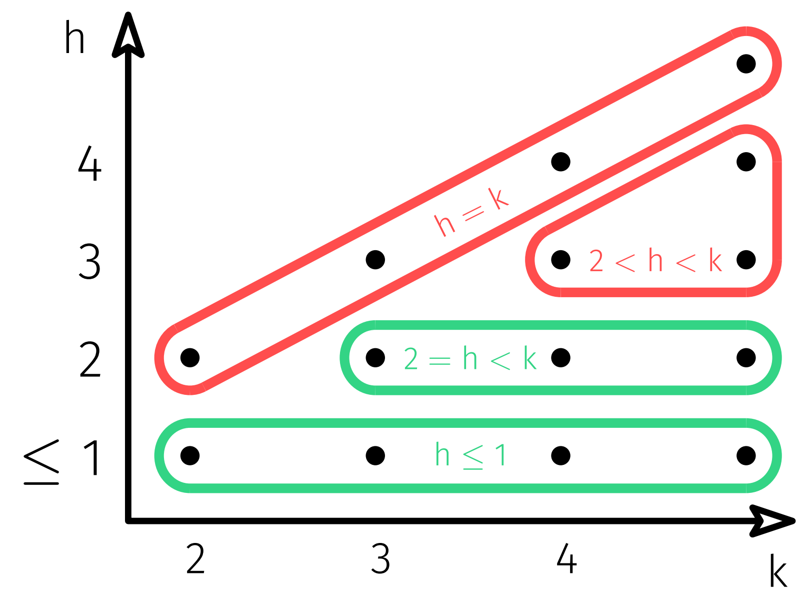


Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

think of x_1, x_2, x_3 as vectors

$\exists x_1 \, \exists x_2 \, \exists x_3 \, \forall y$:
$\varphi(E(x_1, y), E(x_2, y), E(x_3, y))$

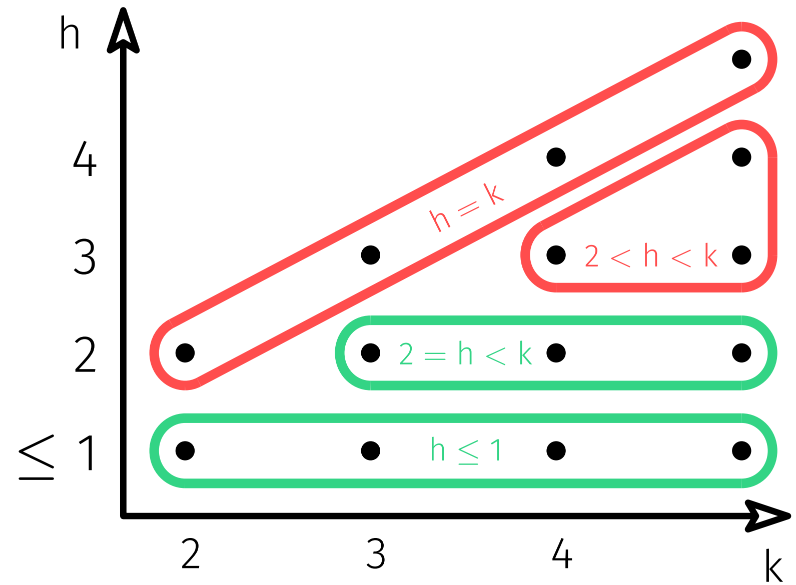


Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

think of
x_1, x_2, x_3
as vectors

$$\exists x_1 \, \exists x_2 \, \exists x_3 \, \forall i: \\ \varphi(x_1[i], x_2[i], x_3[i])$$

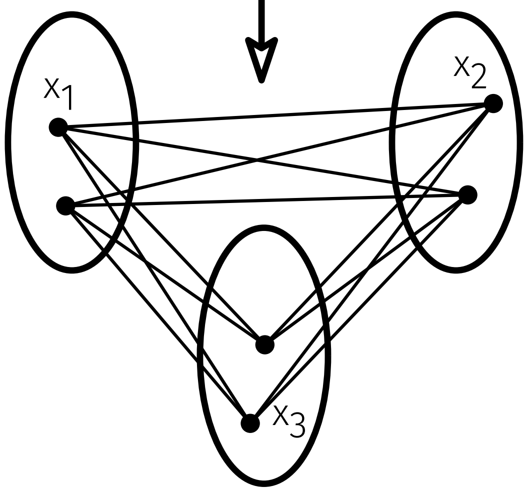


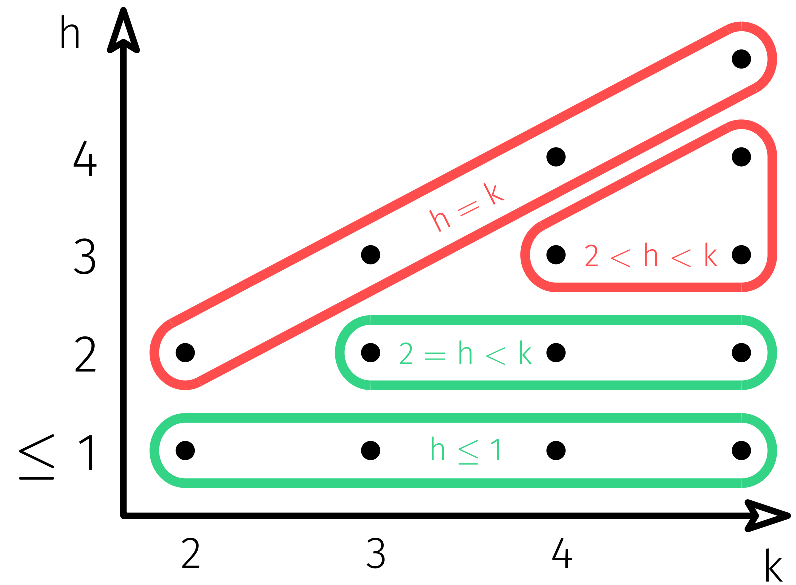
Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

think of x_1, x_2, x_3 as vectors

$\exists x_1 \; \exists x_2 \; \exists x_3 \; \forall i$:
$\varphi(x_1[i], x_2[i], x_3[i])$



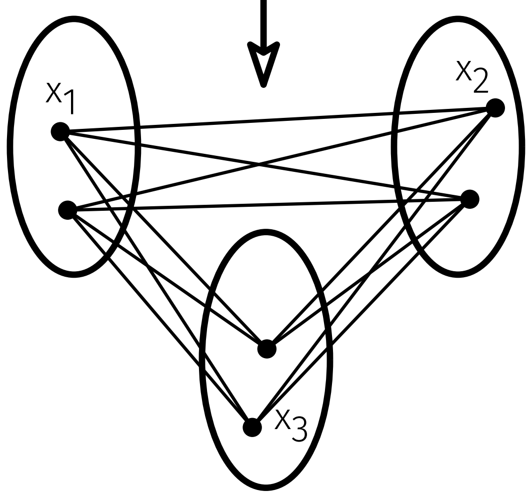


Algorithms for Properties of Hardness $h \leq 2$

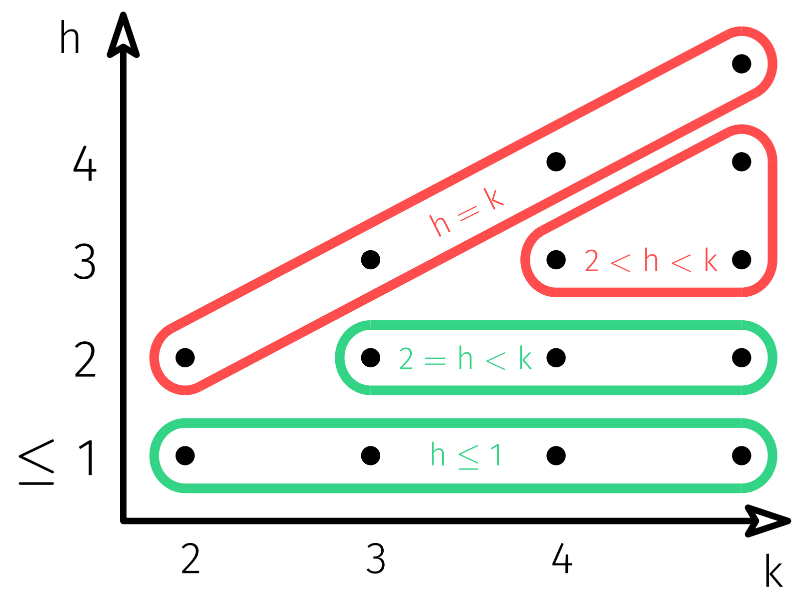
k = 3: Reduction to Constrained Triangles

think of x_1, x_2, x_3 as vectors

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:\ \varphi(x_1[i], x_2[i], x_3[i])$$



O(m) vertices

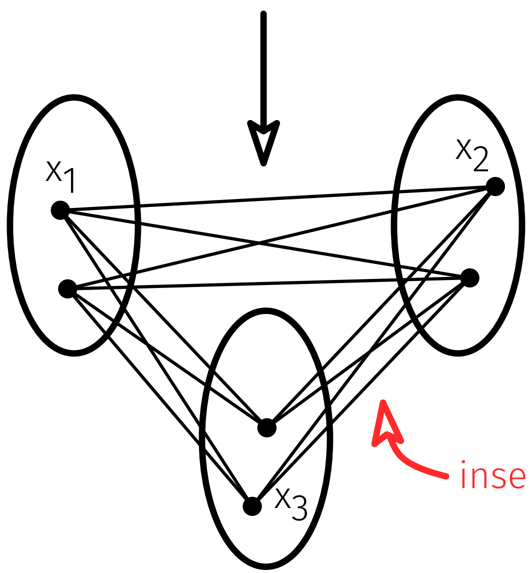


Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

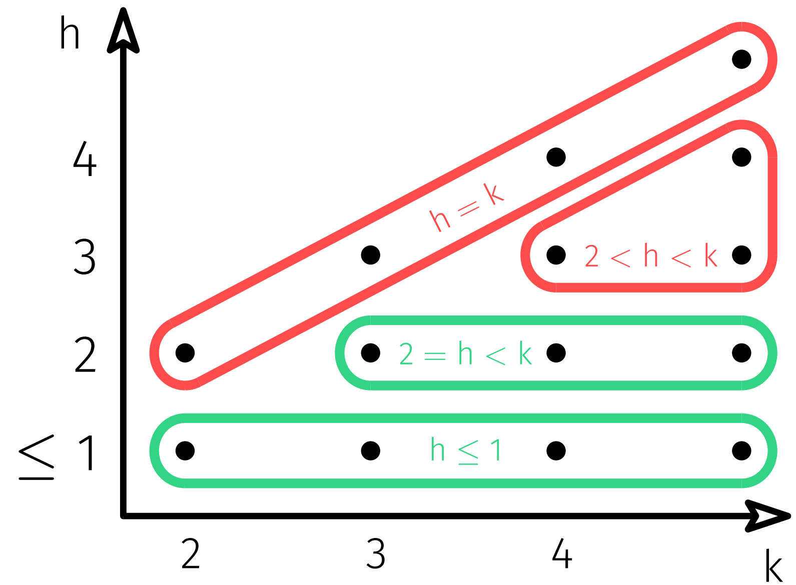
think of
x_1, x_2, x_3
as vectors

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i: \varphi(x_1[i], x_2[i], x_3[i])$$



O(m) vertices

insert all edges

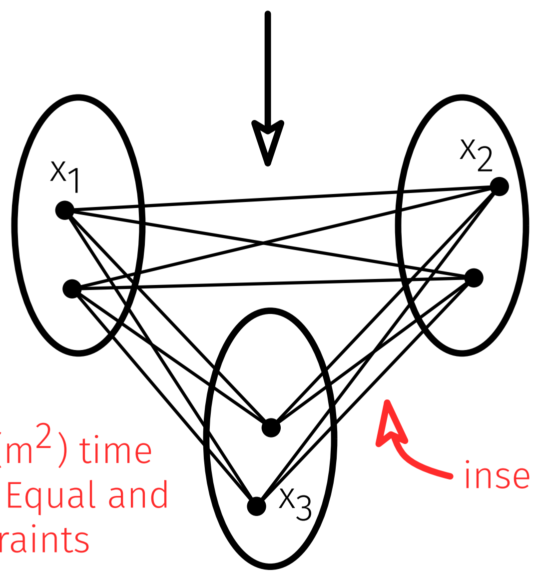


Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

think of x_1, x_2, x_3 as vectors

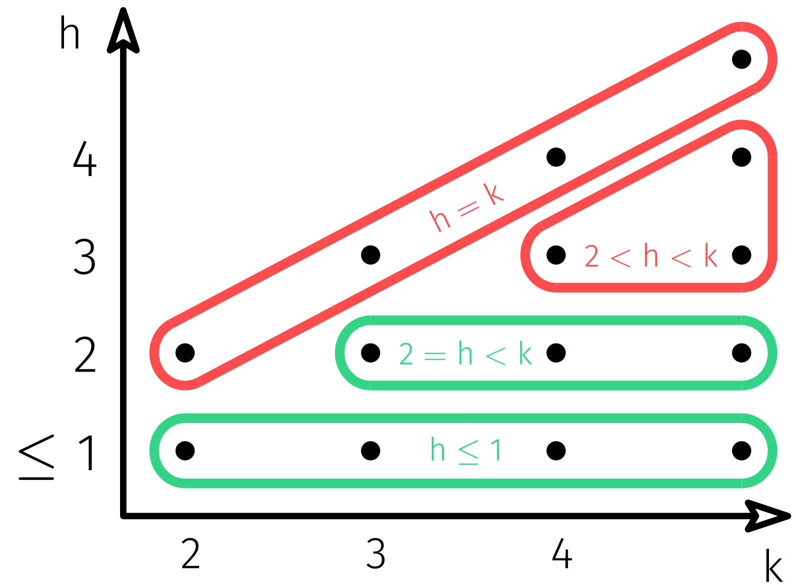
$$\exists x_1 \, \exists x_2 \, \exists x_3 \, \forall i: \varphi(x_1[i], x_2[i], x_3[i])$$



O(m) vertices

insert all edges

Idea: spend $O(m^2)$ time to encode φ by Equal and Sum constraints

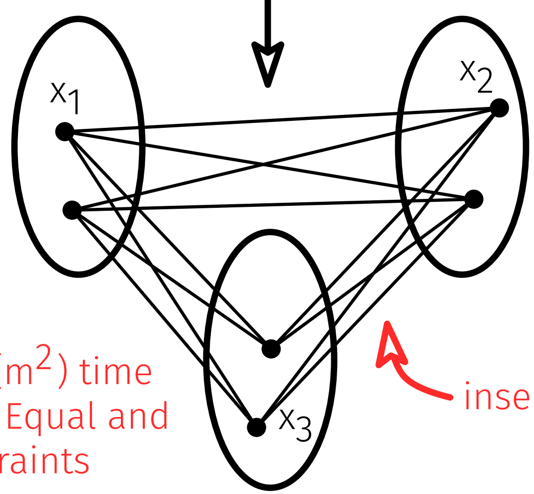


Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

think of
x_1, x_2, x_3
as vectors

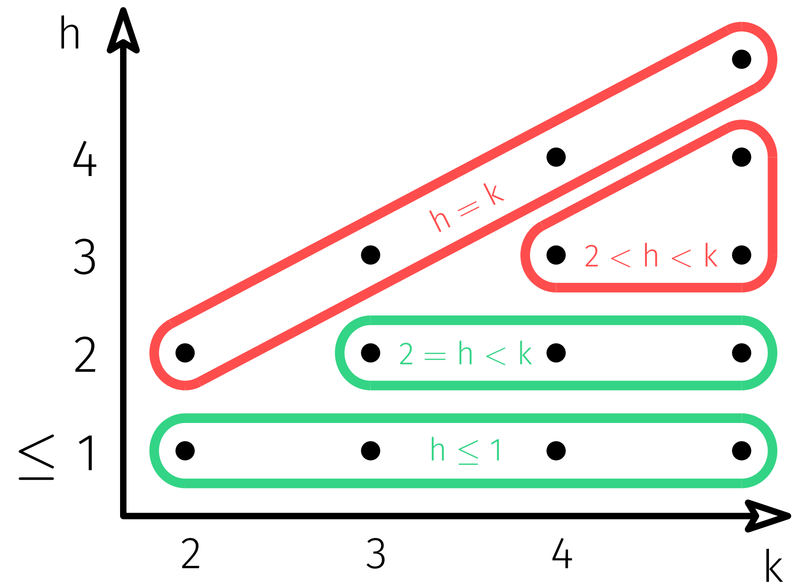
$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i: \varphi(x_1[i], x_2[i], x_3[i])$$



O(m) vertices

insert all edges

Idea: spend $O(m^2)$ time
to encode φ by Equal and
Sum constraints



$k > 3$: Brute-force $k - 3$ quantifiers

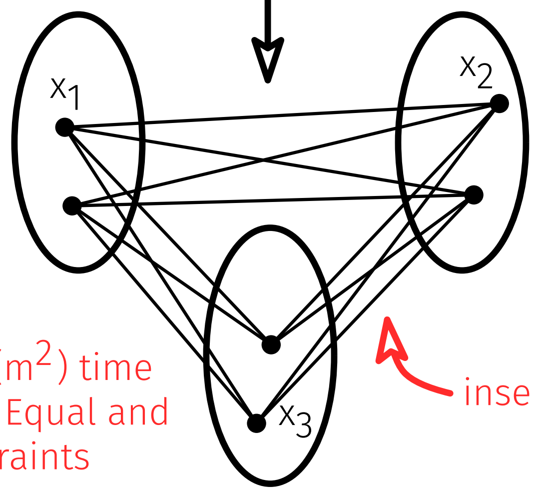
$$\exists x_1\ \exists x_2\ \exists x_3\ \exists x_4\ \forall i: \varphi(x_1[i], x_2[i], x_3[i], x_4[i])$$

Algorithms for Properties of Hardness $h \leq 2$

$\mathbf{k = 3}$: Reduction to Constrained Triangles

think of x_1, x_2, x_3 as vectors

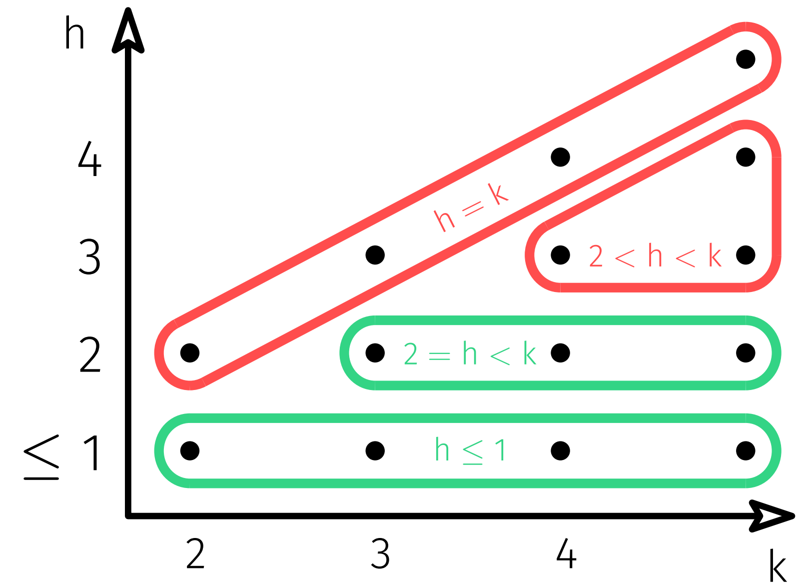
$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$\varphi(x_1[i], x_2[i], x_3[i])$$



O(m) vertices

insert all edges

Idea: spend $O(m^2)$ time to encode φ by Equal and Sum constraints



$\mathbf{k > 3}$: Brute-force $k - 3$ quantifiers

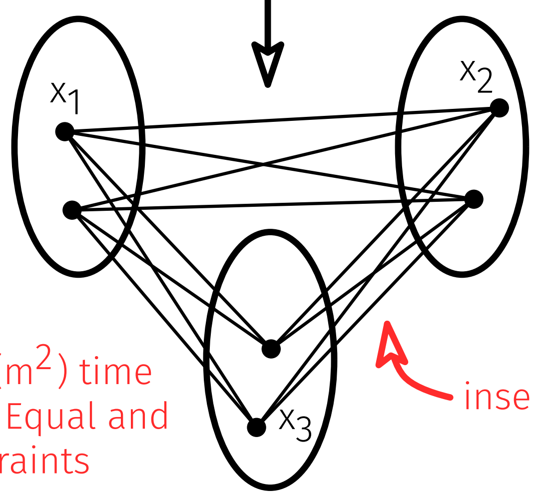
$$\cancel{\exists x_1}\ \exists x_2\ \exists x_3\ \exists x_4\ \forall i:$$
$$\varphi(0/1, x_2[i], x_3[i], x_4[i])$$

Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

think of x_1, x_2, x_3 as vectors

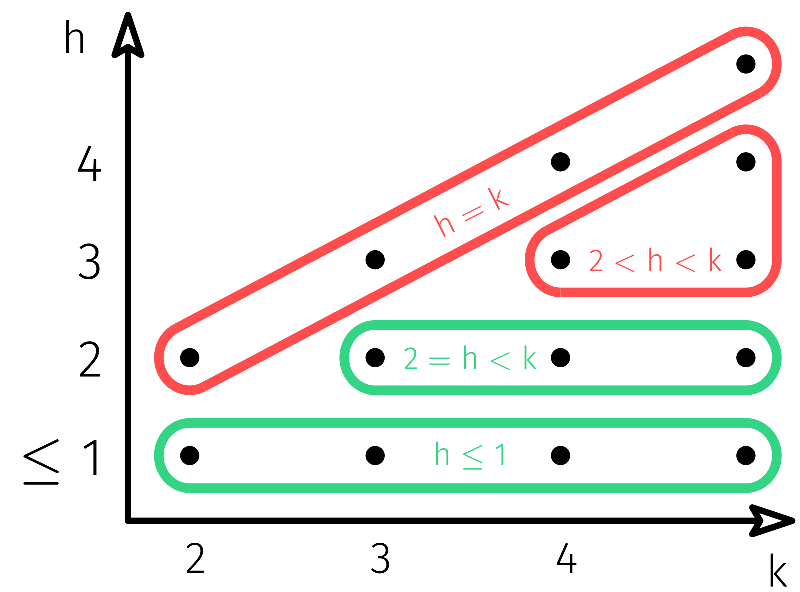
$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:\ \varphi(x_1[i], x_2[i], x_3[i])$$



O(m) vertices

insert all edges

Idea: spend $O(m^2)$ time to encode φ by Equal and Sum constraints



$k > 3$: Brute-force $k - 3$ quantifiers

$$\cancel{\exists x_1}\ \exists x_2\ \exists x_3\ \exists x_4\ \forall i:\ \varphi(0/1, x_2[i], x_3[i], x_4[i])$$

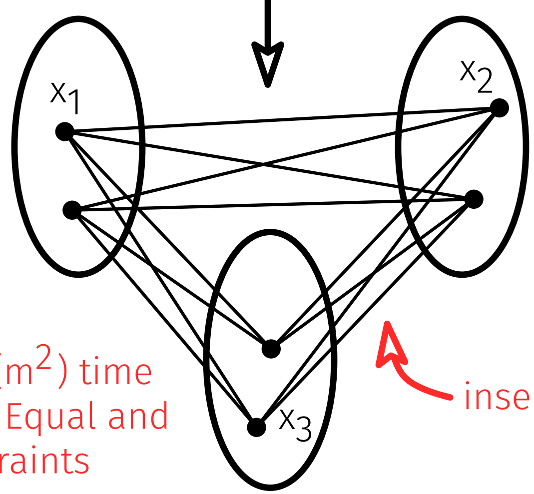
Idea: repeat the above reduction and combine the triangle constraints

Algorithms for Properties of Hardness $h \leq 2$

$\mathbf{k = 3}$: Reduction to Constrained Triangles

think of
x_1, x_2, x_3
as vectors

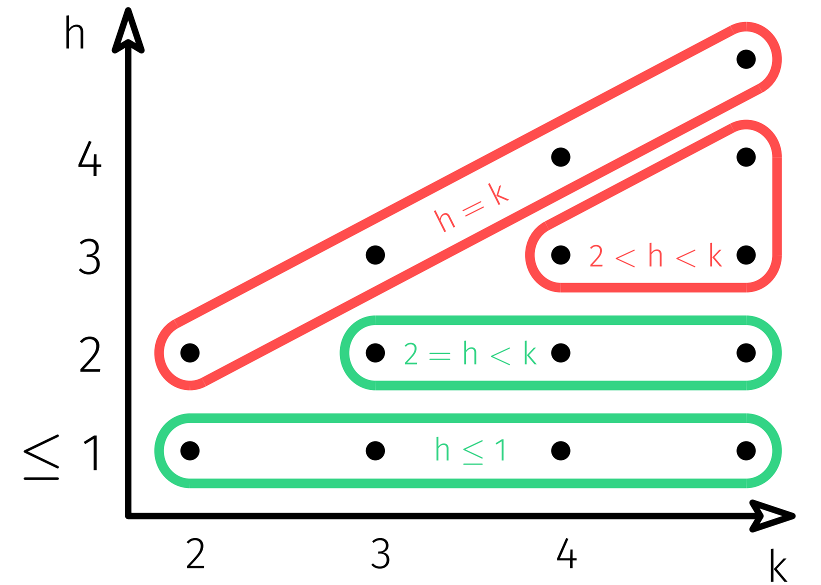
$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$\varphi(x_1[i], x_2[i], x_3[i])$$



O(m) vertices

insert all edges

Idea: spend $O(m^2)$ time
to encode φ by Equal and
Sum constraints



Examples

$\mathbf{k > 3}$: Brute-force $k - 3$ quantifiers

$$\cancel{\exists x_1}\ \exists x_2\ \exists x_3\ \exists x_4\ \forall i:$$
$$\varphi(0/1, x_2[i], x_3[i], x_4[i])$$

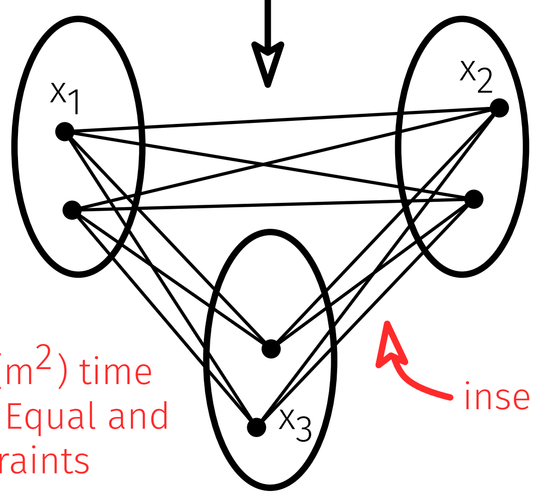
Idea: repeat the above
reduction and combine the
triangle constraints

Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

think of x_1, x_2, x_3 as vectors

$$\exists x_1 \; \exists x_2 \; \exists x_3 \; \forall i: \quad \varphi(x_1[i], x_2[i], x_3[i])$$



O(m) vertices

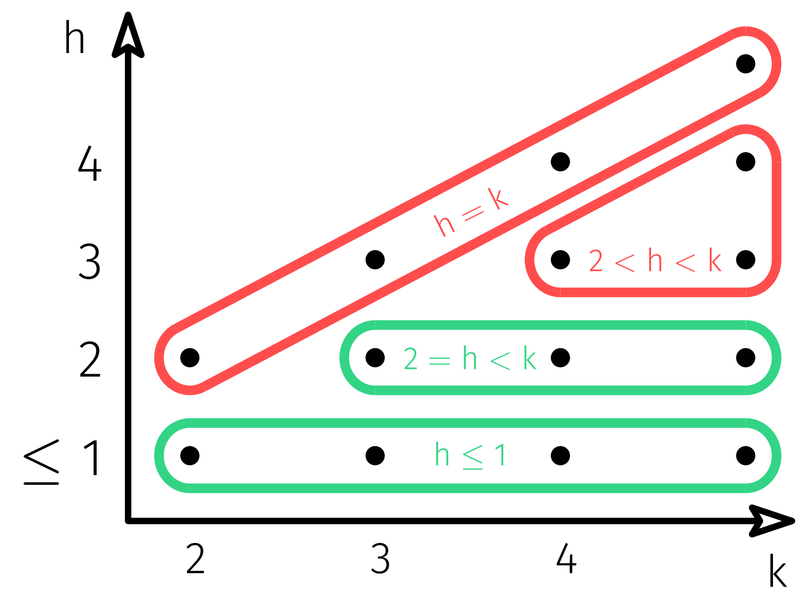
insert all edges

Idea: spend $O(m^2)$ time to encode φ by Equal and Sum constraints

$k > 3$: Brute-force $k - 3$ quantifiers

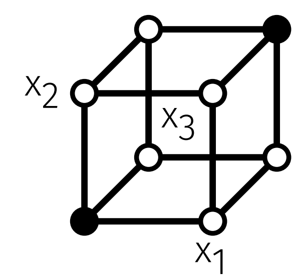
$$\cancel{\exists x_1} \; \exists x_2 \; \exists x_3 \; \exists x_4 \; \forall i: \quad \varphi(0/1, x_2[i], x_3[i], x_4[i])$$

Idea: repeat the above reduction and combine the triangle constraints



Examples

$$\exists x_1 \; \exists x_2 \; \exists x_3 \; \forall i: \quad \mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

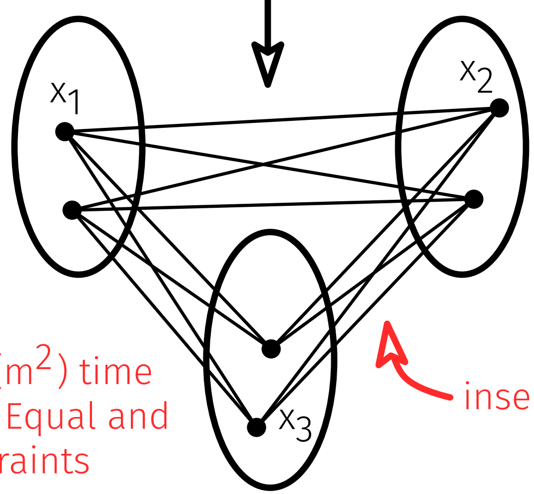


Algorithms for Properties of Hardness $h \leq 2$

k = 3: Reduction to Constrained Triangles

think of x_1, x_2, x_3 as vectors

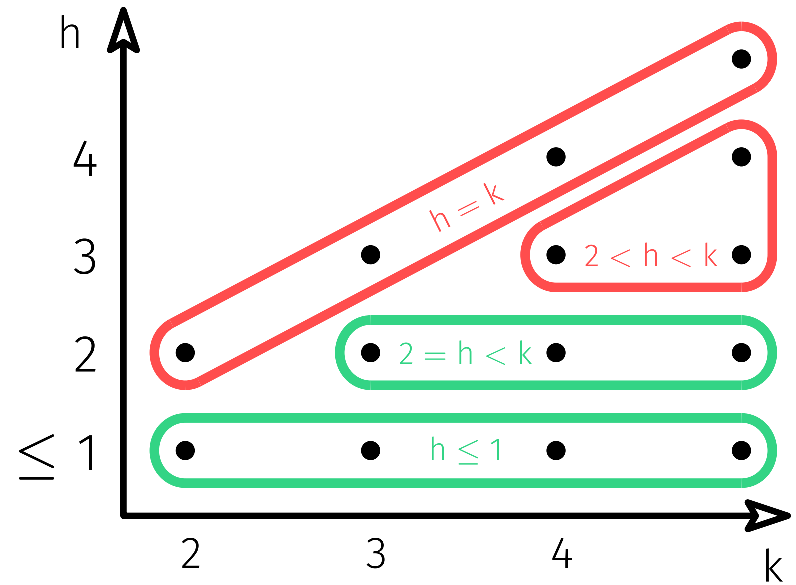
$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i: \varphi(x_1[i], x_2[i], x_3[i])$$



O(m) vertices

insert all edges

Idea: spend $O(m^2)$ time to encode φ by Equal and Sum constraints



k > 3: Brute-force $k-3$ quantifiers

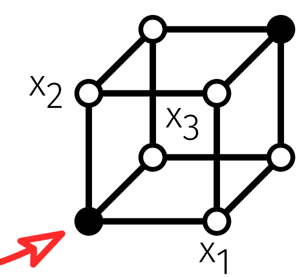
$$\cancel{\exists x_1}\ \exists x_2\ \exists x_3\ \exists x_4\ \forall i: \varphi(0/1, x_2[i], x_3[i], x_4[i])$$

Idea: repeat the above reduction and combine the triangle constraints

Examples

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i: \mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

falsifying assignments

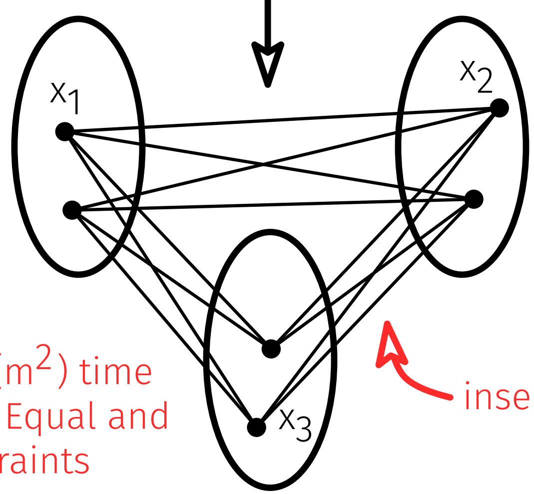


Algorithms for Properties of Hardness $h \leq 2$

$\mathbf{k = 3}$: Reduction to Constrained Triangles

think of x_1, x_2, x_3 as vectors

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i: \varphi(x_1[i], x_2[i], x_3[i])$$



O(m) vertices

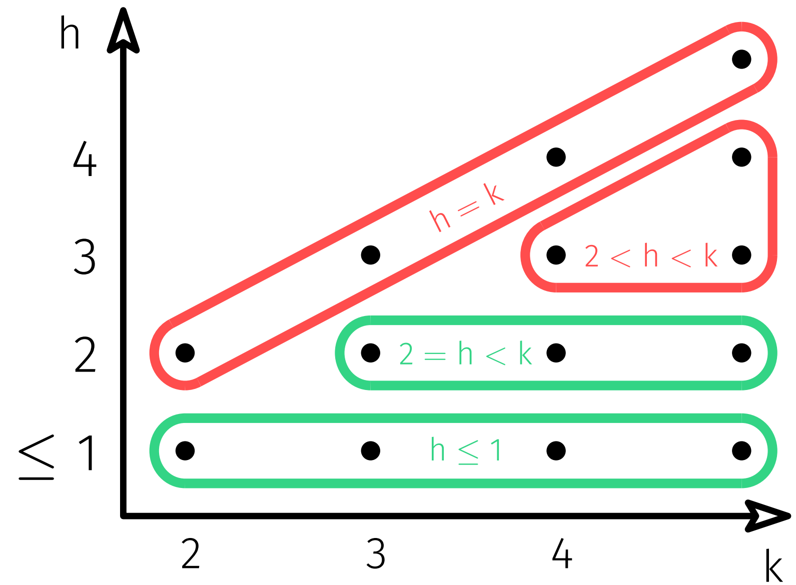
insert all edges

Idea: spend $O(m^2)$ time to encode φ by Equal and Sum constraints

$\mathbf{k > 3}$: Brute-force $k - 3$ quantifiers

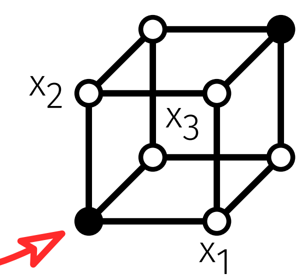
$$\cancel{\exists x_1}\ \exists x_2\ \exists x_3\ \exists x_4\ \forall i: \varphi(0/1, x_2[i], x_3[i], x_4[i])$$

Idea: repeat the above reduction and combine the triangle constraints



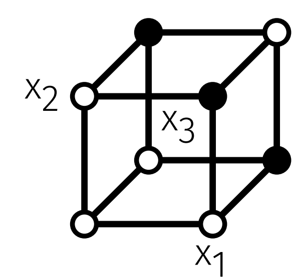
Examples

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i: \mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

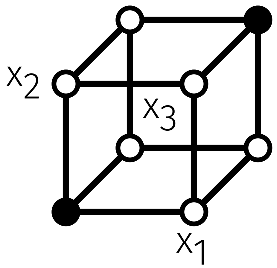


falsifying assignments

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i: x_1[i] + x_2[i] + x_3[i] \neq 2$$

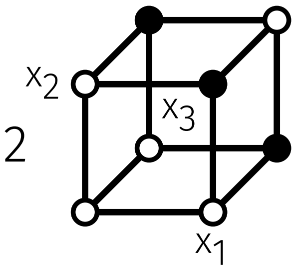


How to Employ Sum and Equal Constraints

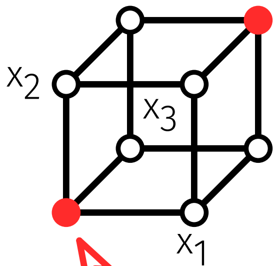


$$\exists x_1 \;\; \exists x_2 \;\; \exists x_3 \;\; \forall i:$$
$$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

$$\exists x_1 \;\; \exists x_2 \;\; \exists x_3 \;\; \forall i:$$
$$x_1[i] + x_2[i] + x_3[i] \neq 2$$



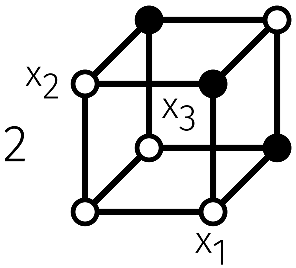
How to Employ Sum and Equal Constraints



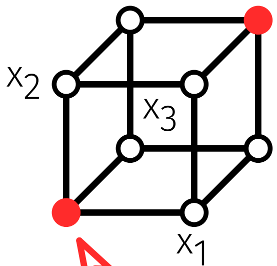
$$\exists x_1 \ \exists x_2 \ \exists x_3 \ \forall i:$$
$$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

Idea: Exclude 000 and 111 by a Sum constraint

$$\exists x_1 \ \exists x_2 \ \exists x_3 \ \forall i:$$
$$x_1[i] + x_2[i] + x_3[i] \neq 2$$



How to Employ Sum and Equal Constraints



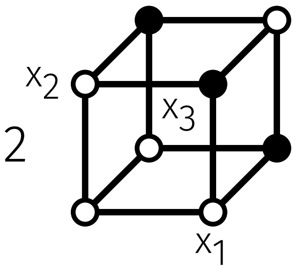
$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$
$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$

Idea: Exclude 000 and 111 by a Sum constraint

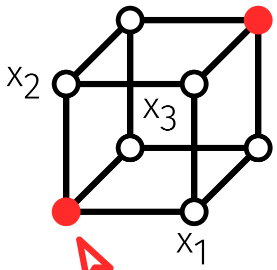
Exclude 111 and 000 in all dimensions

$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$

$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$
$x_1[i] + x_2[i] + x_3[i] \neq 2$



How to Employ Sum and Equal Constraints



$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$
$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$

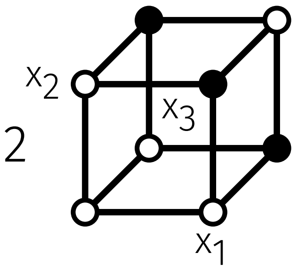
Idea: Exclude 000 and 111 by a Sum constraint

Exclude 111 and 000 in all dimensions

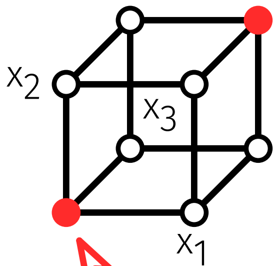
$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$

$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$

$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$
$x_1[i] + x_2[i] + x_3[i] \neq 2$



How to Employ Sum and Equal Constraints



$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

Idea: Exclude 000 and 111 by a Sum constraint

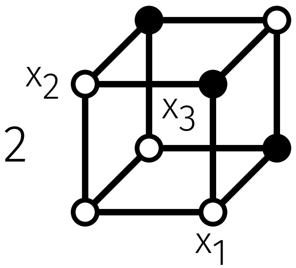
Exclude 111 and 000 in all dimensions

$$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

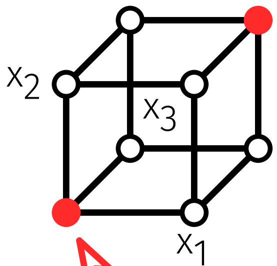
$$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\begin{aligned}\Leftrightarrow\ 0 &= \|x_1 \wedge x_2 \wedge x_3\|_1 - \|x_1 \wedge x_2 \wedge x_3\|_1 \\ &+ \|x_1 \wedge x_2\|_1 - \|x_1\|_1 \\ &+ \|x_2 \wedge x_3\|_1 - \|x_2\|_1 \\ &+ \|x_3 \wedge x_1\|_1 - \|x_3\|_1 \\ &+ d\end{aligned}$$

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$x_1[i] + x_2[i] + x_3[i] \neq 2$$



How to Employ Sum and Equal Constraints



$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

Idea: Exclude 000 and 111 by a Sum constraint

Exclude 111 and 000 in all dimensions

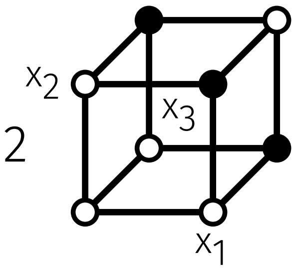
$$\Leftrightarrow\quad 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow\quad 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

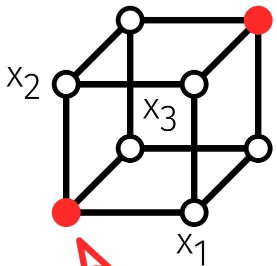
$$\begin{aligned}\Leftrightarrow\quad 0 = & \boxed{\|x_1 \wedge x_2 \wedge x_3\|_1 - \|x_1 \wedge x_2 \wedge x_3\|_1} \\ & + \|x_1 \wedge x_2\|_1 - \|x_1\|_1 \\ & + \|x_2 \wedge x_3\|_1 - \|x_2\|_1 \\ & + \|x_3 \wedge x_1\|_1 - \|x_3\|_1 \\ & + d\end{aligned}$$

cancels!

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$x_1[i] + x_2[i] + x_3[i] \neq 2$$



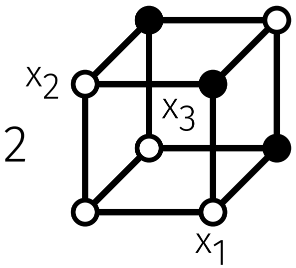
How to Employ Sum and Equal Constraints



$$\exists x_1 \;\exists x_2 \;\exists x_3 \;\forall i:$$
$$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

Idea: Exclude 000 and 111 by a Sum constraint

$$\exists x_1 \;\exists x_2 \;\exists x_3 \;\forall i:$$
$$x_1[i] + x_2[i] + x_3[i] \neq 2$$



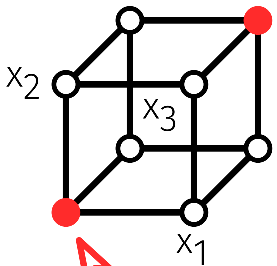
Exclude 111 and 000 in all dimensions

$$\Leftrightarrow \quad 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow \quad 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\begin{aligned}
\Leftrightarrow \quad 0 = &\boxed{\|x_1 \wedge x_2 \wedge x_3\|_1 - \|x_1 \wedge x_2 \wedge x_3\|_1} \quad \text{cancels!} \\
&+ \boxed{\|x_1 \wedge x_2\|_1 - \|x_1\|_1} \; w(x_1, x_2) \\
&+ \boxed{\|x_2 \wedge x_3\|_1 - \|x_2\|_1} \; w(x_2, x_3) \\
&+ \boxed{\|x_3 \wedge x_1\|_1 - \|x_3\|_1} \; w(x_3, x_1) \\
&+ \boxed{d} \; \text{target } t
\end{aligned}$$

How to Employ Sum and Equal Constraints



$\exists x_1 \ \exists x_2 \ \exists x_3 \ \forall i$:
$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$

Idea: Exclude 000
and 111 by a Sum
constraint

Exclude 111 and 000 in all dimensions

$$\Leftrightarrow \quad 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow \quad 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow \quad 0 = \boxed{\|x_1 \wedge x_2 \wedge x_3\|_1 - \|x_1 \wedge x_2 \wedge x_3\|_1}$$

cancels!

$$+ \boxed{\|x_1 \wedge x_2\|_1 - \|x_1\|_1} \ w(x_1, x_2)$$

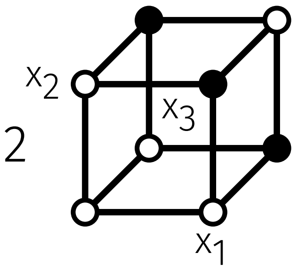
$$+ \boxed{\|x_2 \wedge x_3\|_1 - \|x_2\|_1} \ w(x_2, x_3)$$

$$+ \boxed{\|x_3 \wedge x_1\|_1 - \|x_3\|_1} \ w(x_3, x_1)$$

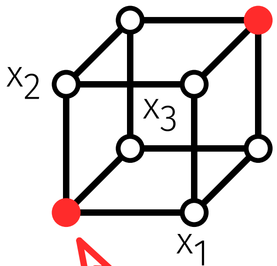
$$+ \boxed{d} \ \text{target } t$$

Generalizes for any pair of falsifying assignments of odd Hamming distance

$\exists x_1 \ \exists x_2 \ \exists x_3 \ \forall i$:
$x_1[i] + x_2[i] + x_3[i] \neq 2$



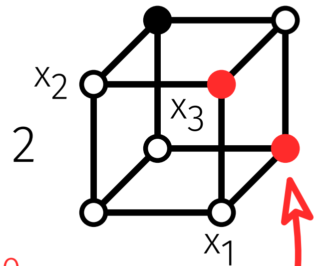
How to Employ Sum and Equal Constraints



$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

Idea: Exclude 000
and 111 by a Sum
constraint

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$x_1[i] + x_2[i] + x_3[i] \neq 2$$



Idea: Exclude 110
and 101 by an Equal
constraint

Exclude 111 and 000 in all dimensions

$$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow\ 0 = \boxed{\|x_1 \wedge x_2 \wedge x_3\|_1 - \|x_1 \wedge x_2 \wedge x_3\|_1}$$

cancels!

$$+ \boxed{\|x_1 \wedge x_2\|_1 - \|x_1\|_1}\ w(x_1, x_2)$$

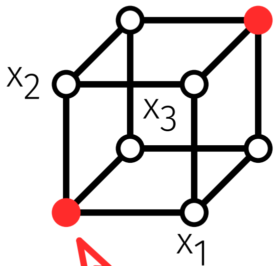
$$+ \boxed{\|x_2 \wedge x_3\|_1 - \|x_2\|_1}\ w(x_2, x_3)$$

$$+ \boxed{\|x_3 \wedge x_1\|_1 - \|x_3\|_1}\ w(x_3, x_1)$$

$$+ \boxed{d}\ \text{target } t$$

Generalizes for any pair of falsifying
assignments of odd Hamming distance

How to Employ Sum and Equal Constraints



$$\exists x_1 \ \exists x_2 \ \exists x_3 \ \forall i:$$
$$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

Idea: Exclude 000 and 111 by a Sum constraint

Exclude 111 and 000 in all dimensions

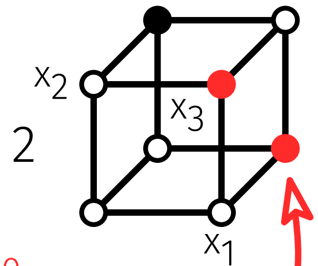
$$\Leftrightarrow \quad 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow \quad 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow \quad 0 = \underbrace{\boxed{\|x_1 \wedge x_2 \wedge x_3\|_1 - \|x_1 \wedge x_2 \wedge x_3\|_1}}_{\text{cancels!}}$$
$$+ \boxed{\|x_1 \wedge x_2\|_1 - \|x_1\|_1} \ w(x_1, x_2)$$
$$+ \boxed{\|x_2 \wedge x_3\|_1 - \|x_2\|_1} \ w(x_2, x_3)$$
$$+ \boxed{\|x_3 \wedge x_1\|_1 - \|x_3\|_1} \ w(x_3, x_1)$$
$$+ \boxed{d} \ \text{target } t$$

Generalizes for any pair of falsifying assignments of odd Hamming distance

$$\exists x_1 \ \exists x_2 \ \exists x_3 \ \forall i:$$
$$x_1[i] + x_2[i] + x_3[i] \neq 2$$

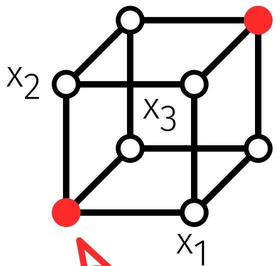


Idea: Exclude 110 and 101 by an Equal constraint

Exclude 110 and 101 in all dimensions

$$\Leftrightarrow \quad x_1 \wedge x_2 \ = \ x_1 \wedge x_3$$

How to Employ Sum and Equal Constraints



$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

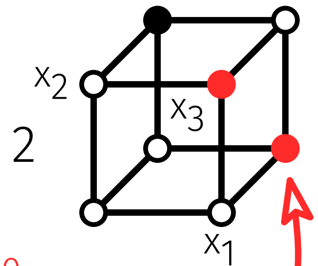
Idea: Exclude 000 and 111 by a Sum constraint

Exclude 111 and 000 in all dimensions

$$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$
$$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$
$$\begin{aligned}\Leftrightarrow\ 0 = {} & \boxed{\|x_1 \wedge x_2 \wedge x_3\|_1 - \|x_1 \wedge x_2 \wedge x_3\|_1} \text{ cancels!}\\ & + \boxed{\|x_1 \wedge x_2\|_1 - \|x_1\|_1}\ w(x_1, x_2)\\ & + \boxed{\|x_2 \wedge x_3\|_1 - \|x_2\|_1}\ w(x_2, x_3)\\ & + \boxed{\|x_3 \wedge x_1\|_1 - \|x_3\|_1}\ w(x_3, x_1)\\ & + \boxed{d}\ \text{target } t\end{aligned}$$

Generalizes for any pair of falsifying assignments of odd Hamming distance

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$x_1[i] + x_2[i] + x_3[i] \neq 2$$

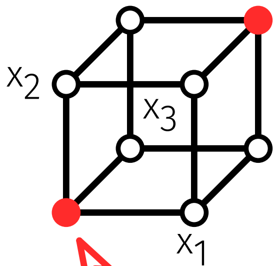


Idea: Exclude 110 and 101 by an Equal constraint

Exclude 110 and 101 in all dimensions

$$\Leftrightarrow\ \underbrace{\boxed{x_1 \wedge x_2}}_{w(x_1, x_2)} = \underbrace{\boxed{x_1 \wedge x_3}}_{w(x_1, x_3)}$$

How to Employ Sum and Equal Constraints



$$\exists x_1 \ \exists x_2 \ \exists x_3 \ \forall i: \ \mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

Idea: Exclude 000 and 111 by a Sum constraint

Exclude 111 and 000 in all dimensions

$$\Leftrightarrow \ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow \ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow \ 0 = \boxed{\|x_1 \wedge x_2 \wedge x_3\|_1 - \|x_1 \wedge x_2 \wedge x_3\|_1}$$

cancels!

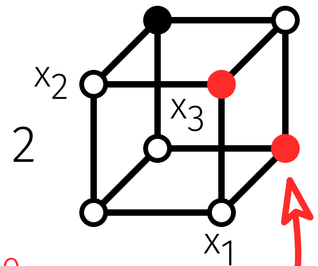
$$+ \boxed{\|x_1 \wedge x_2\|_1 - \|x_1\|_1} \ w(x_1, x_2)$$

$$+ \boxed{\|x_2 \wedge x_3\|_1 - \|x_2\|_1} \ w(x_2, x_3)$$

$$+ \boxed{\|x_3 \wedge x_1\|_1 - \|x_3\|_1} \ w(x_3, x_1)$$

$$+ \boxed{d} \ \text{target } t$$

Generalizes for any pair of falsifying assignments of odd Hamming distance



$$\exists x_1 \ \exists x_2 \ \exists x_3 \ \forall i: \ x_1[i] + x_2[i] + x_3[i] \neq 2$$

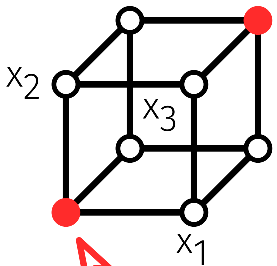
Idea: Exclude 110 and 101 by an Equal constraint

Exclude 110 and 101 in all dimensions

$$\Leftrightarrow \ \underbrace{\boxed{x_1 \wedge x_2}}_{w(x_1, x_2)} = \underbrace{\boxed{x_1 \wedge x_3}}_{w(x_1, x_3)}$$

convert vectors arbitrarily into numbers

How to Employ Sum and Equal Constraints



$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$\mathrm{NAE}(x_1[i], x_2[i], x_3[i])$$

Idea: Exclude 000
and 111 by a Sum
constraint

Exclude 111 and 000 in all dimensions

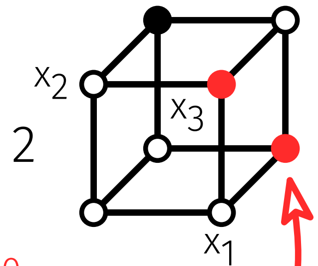
$$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 = \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\Leftrightarrow\ 0 = \|x_1 \wedge x_2 \wedge x_3\|_1 + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|_1$$

$$\begin{aligned}\Leftrightarrow\ 0 = {} & \boxed{\|x_1 \wedge x_2 \wedge x_3\|_1 - \|x_1 \wedge x_2 \wedge x_3\|_1} \text{ cancels!} \\ & + \boxed{\|x_1 \wedge x_2\|_1 - \|x_1\|_1}\ w(x_1, x_2) \\ & + \boxed{\|x_2 \wedge x_3\|_1 - \|x_2\|_1}\ w(x_2, x_3) \\ & + \boxed{\|x_3 \wedge x_1\|_1 - \|x_3\|_1}\ w(x_3, x_1) \\ & + \boxed{d} \text{ target } t\end{aligned}$$

Generalizes for any pair of falsifying
assignments of odd Hamming distance

$$\exists x_1\ \exists x_2\ \exists x_3\ \forall i:$$
$$x_1[i] + x_2[i] + x_3[i] \neq 2$$



Idea: Exclude 110
and 101 by an Equal
constraint

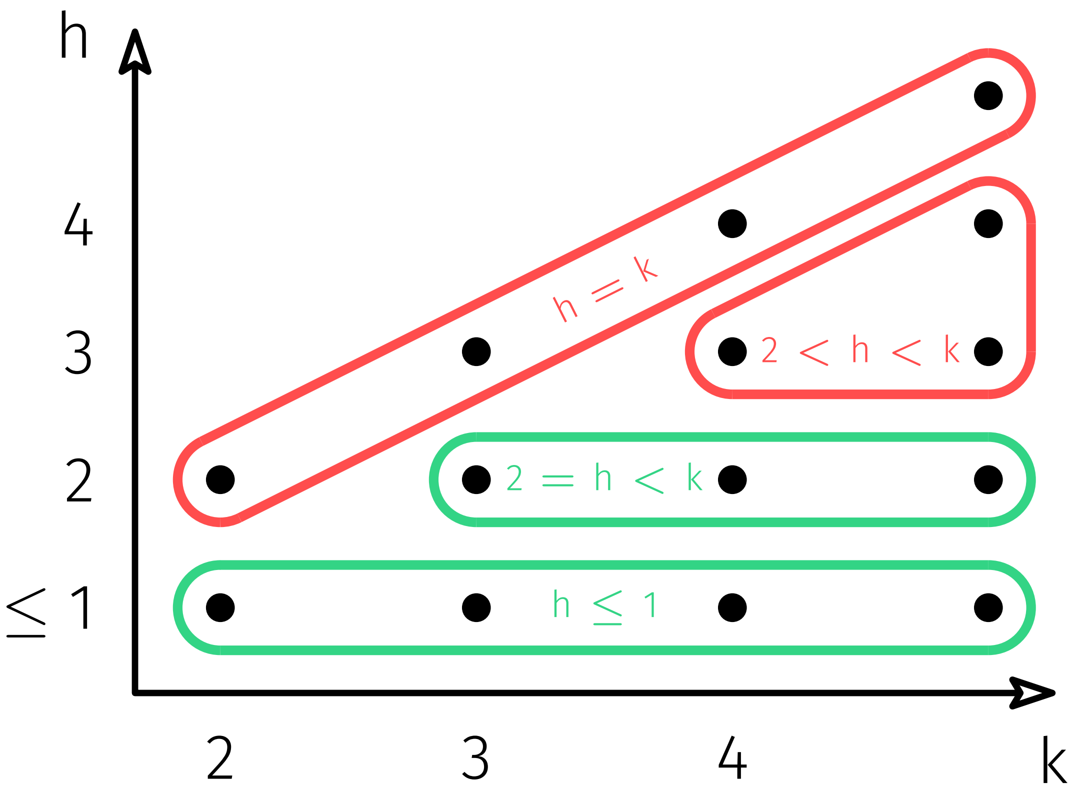
Exclude 110 and 101 in all dimensions

$$\Leftrightarrow\ \underbrace{\boxed{x_1 \wedge x_2}}_{w(x_1, x_2)} = \underbrace{\boxed{x_1 \wedge x_3}}_{w(x_1, x_3)}$$

convert vectors arbitrarily
into numbers

Generalizes for any pair of falsifying
assignments of even Hamming distance

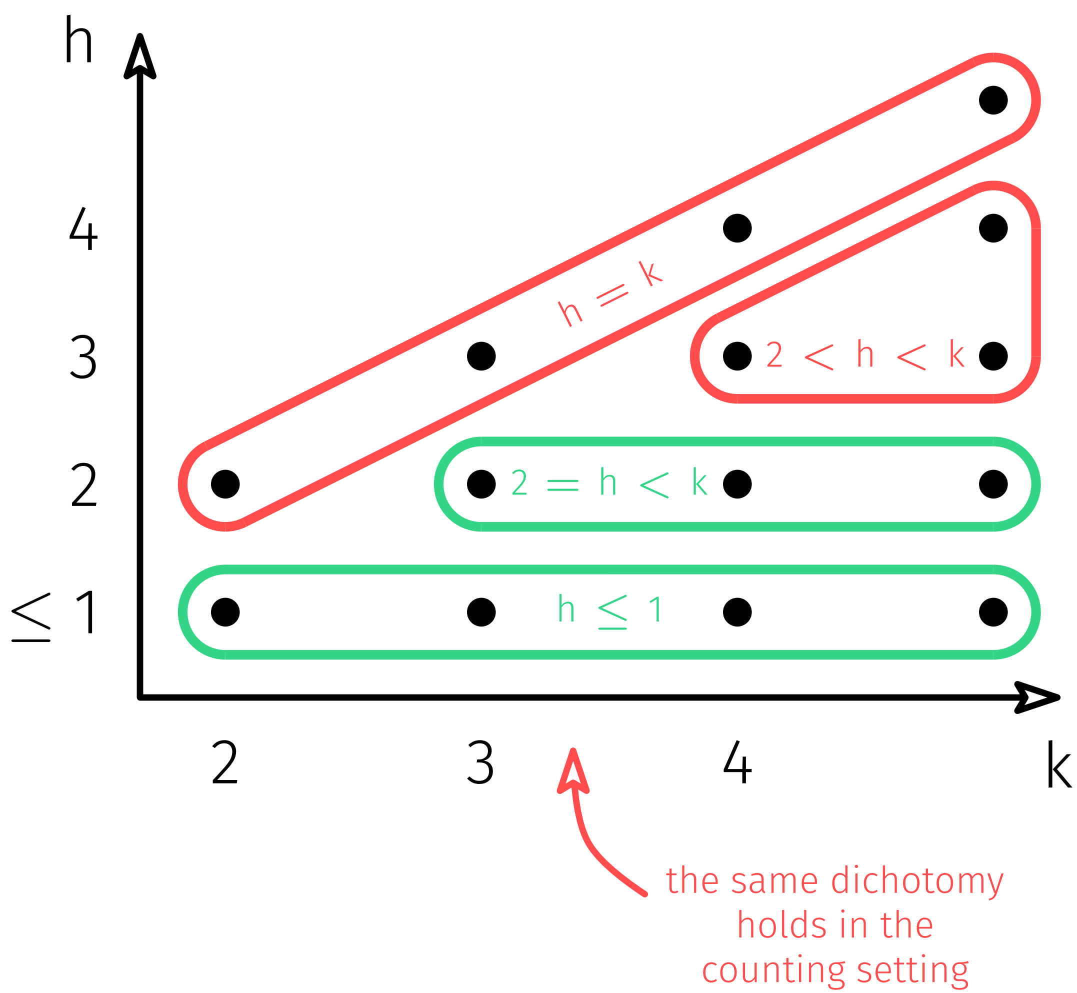
Conclusion and Open Problems



Open problems

- In which way does the classification extend to first-order queries beyond $\exists^k\forall$-quantified graph properties?
- What's the exact complexity of low-hardness properties?
- Equivalence of finding cliques in h-hypergraphs and properties of hardness h?

Conclusion and Open Problems



Open problems

- In which way does the classification extend to first-order queries beyond $\exists^k\forall$-quantified graph properties?
- What's the exact complexity of low-hardness properties?
- Equivalence of finding cliques in h-hypergraphs and properties of hardness h?