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 Künemann$^1$

$^1$ Max Planck Institute for Informatics, Saarland Informatics Campus (SIC), Saarbrücken
First-Order Property Model-Checking

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

$$\exists x \exists y \exists z: E(x, y) \land E(y, z) \land E(z, x)$$
Fix a first-order property $\psi$. The model-checking problem for $\psi$ asks to check whether $\psi$ is true on a given structure (e.g. graph).

$$\exists x \exists y \exists z: E(x, y) \land E(y, z) \land E(z, x)$$
Fix a first-order property $\psi$. The model-checking problem for $\psi$ asks to check whether $\psi$ is true on a given structure (e.g. graph).

$\exists x \exists y \exists z: E(x, y) \land E(y, z) \land E(z, x)$

3-independent set
Fix a first-order property $\psi$. The model-checking problem for $\psi$ asks to check whether $\psi$ is true on a given structure (e.g. graph).

$$\exists x \exists y \exists z: E(x, y) \land E(y, z) \land E(z, x)$$

3-independent set

### SQL

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fix a first-order property $\psi$. The model-checking problem for $\psi$ asks to check whether $\psi$ is true on a given structure (e.g. graph).

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

3-independent set

we measure the complexity in the number of edges $m$

SQL

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fix a first-order property $\psi$. The model-checking problem for $\psi$ asks to check whether $\psi$ is true on a given structure (e.g. graph).

\[
\exists x \exists y \exists z: E(x, y) \land E(y, z) \land E(z, x)
\]

3-independent set

we measure the complexity in the number of edges $m$

SQL

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

here, $m$ equals the size of the database
Orthogonal Vectors

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

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

here, $m$ equals the total number of 1-entries

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

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

$O(m^k)$ 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

<table>
<thead>
<tr>
<th>Constraint satisfaction problems</th>
<th>First-order properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-SAT is NP-complete</td>
<td>k-OV is FOP-complete</td>
</tr>
<tr>
<td>[Cook '71]</td>
<td>[GIKW '17]</td>
</tr>
<tr>
<td>Every Boolean CSP is either in P or NP-complete [Schaefer '78]</td>
<td>?</td>
</tr>
</tbody>
</table>
Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties
Our Main Result:
A Classification of $\exists^k \forall$-Quantified Graph Properties

$$\psi = \exists x_1 \ldots \exists x_k \forall y: \phi(E(x_1, y), \ldots, E(x_k, y))$$
Our Main Result:
A Classification of $\exists^k \forall$-Quantified Graph Properties

$\psi = \exists x_1 \ldots \exists x_k \forall y:\varphi(E(x_1, y), \ldots, E(x_k, y))$

Boolean function
$\varphi : \{0, 1\}^k \rightarrow \{0, 1\}$
Our Main Result:
A Classification of \( \exists^k \forall \)-Quantified Graph Properties

\[
\psi = \exists x_1 \ldots \exists x_k \forall y: \varphi(E(x_1, y), \ldots, E(x_k, y))
\]

Boolean function \( \varphi : \{0, 1\}^k \rightarrow \{0, 1\} \)

The hardness of \( \psi \) is the largest number \( h \in \{0, \ldots, k\} \), such that

“\( \psi \) is \( h \)-OV-like”
Our Main Result:
A Classification of $\exists^k \forall$-Quantified Graph Properties

$$\psi = \exists x_1 \ldots \exists x_k \forall y: \varphi(E(x_1, y), \ldots, E(x_k, y))$$

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

The hardness of $\psi$ is the largest number $h \in \{0, \ldots, k\}$, such that for any subset $J \subseteq [k]$ of $k - h$ inputs, there exists an assignment $\alpha : J \rightarrow \{0, 1\}$, so that $\varphi|_{\alpha}$ has exactly one falsifying assignment.
Our Main Result:
A Classification of $\exists^k \forall$-Quantified Graph Properties

$$\psi = \exists x_1 \ldots \exists x_k \forall y: \varphi(E(x_1, y), \ldots, E(x_k, y))$$

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

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

“$\psi$ is $h$-OV-like”
Our Main Result:
A Classification of $\exists^k \forall$-Quantified Graph Properties

$\psi = \exists x_1 \ldots \exists x_k \forall y: \varphi(E(x_1, y), \ldots, E(x_k, y))$

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

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

"$\psi$ is $h$-OV-like"
Our Main Result:
A Classification of $\exists^k \forall$-Quantified Graph Properties

\[ \psi = \exists x_1 \ldots \exists x_k \forall y: \varphi(E(x_1, y), \ldots, E(x_k, y)) \]

Boolean function \( \varphi : \{0, 1\}^k \rightarrow \{0, 1\} \)

The hardness of \( \psi \) is the largest number \( h \in \{0, \ldots, k\} \), such that

“\( \psi \) is \( h \)-OV-like”

require time \( m^{k-o(1)} \) under the Hyperclique hypothesis

decidable in time \( O(m^{k-\epsilon}) \) for some \( \epsilon > 0 \)
Our Main Result: A Classification of $\exists^k \forall$-Quantified Graph Properties

$$\psi = \exists x_1 \ldots \exists x_k \forall y : \varphi(E(x_1, y), \ldots, E(x_k, y))$$

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

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

"$\psi$ is h-OV-like"

require time $m^{k-o(1)}$ under the Hyperclique hypothesis

decidable in time $O(m^{k-\epsilon})$ for some $\epsilon > 0$
Our Main Result:
A Classification of $\exists^k \forall$-Quantified Graph Properties

$\psi = \exists x_1 \ldots \exists x_k \forall y: \varphi(E(x_1, y), \ldots, E(x_k, y))$

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

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

“$\psi$ is $h$-OV-like”

require time $m^k - o(1)$ under SETH [GIKW '17]

require time $m^k - o(1)$ under the Hyperclique hypothesis

decidable in time $O(m^{k-\epsilon})$ for some $\epsilon > 0$

$h \leq 2$ and $h < k$

$h = k$

$2 < h < k$
Our Main Result:
A Classification of $\exists^k\forall$-Quantified Graph Properties

$$\psi = \exists x_1 \ldots \exists x_k \forall y: \phi(E(x_1, y), \ldots, E(x_k, y))$$

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

"$\psi$ is $h$-OV-like"

- Require time $m^{k-o(1)}$ under SETH [GIKW '17]
- Require time $m^{k-o(1)}$ under the Hyperclique hypothesis
- Decidable in time $O(m^{k-\epsilon})$ for some $\epsilon > 0$
Our Main Result:
A Classification of \( \exists^k \forall \)-Quantified Graph Properties

\[
\psi = \exists x_1 \ldots \exists x_k \forall y : \varphi(E(x_1, y), \ldots, E(x_k, y))
\]

The hardness of \( \psi \) is the largest number \( h \in \{0, \ldots, k\} \), such that

“\( \psi \) is h-OV-like”

![Diagram](image-url)

- \( 2 \leq h \leq 1 \) requires time \( m^{k-o(1)} \) under SETH [GIKW '17]
- \( 2 = h < k \) is decidable in time \( O(m^{k-\epsilon}) \) for some \( \epsilon > 0 \)
- \( h = k \) require time \( m^{k-o(1)} \) under the Hyperclique hypothesis
- \( 2 < h < k \) requires fast matrix multiplication
- The speed-up requires 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 $n^{k-o(1)}$. 
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]
**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 $O(2^{(1-\epsilon)n})$ [Williams '07]

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

**Our results:**

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
- is implied by the assumption that MAX-3-SAT cannot be solved in time $O(2^{(1-\varepsilon)n})$ [Williams ’07]
- Strassen-like techniques are ruled out [Lincoln, V-Williams, Williams ’18]

**Our results:** Hardness levels

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

$h = 3$

$h = 4$

$h = k$

$h \leq 2$

hard under SETH
Build your own cubic problem!

**Step 1:** Take the basis problem

\[ \Theta(n^3) \quad O(n^\omega) \]

(Triangle Detection)

**Step 2:** Choose your toppings

\[ \Theta(n^3) \quad O(n^3 - \epsilon) \]

(Equal Constraint)

(Sum Constraint)
Build your own cubic problem!

**Step 1:** Take the basis problem
\[ \Theta(n^3) O(n^\omega) \]

(Triangle Detection)

**Step 2:** Choose your toppings
\[ \Theta(n^3) O(n^{3-\varepsilon}) \]

(Equal Constraint)

we assume tripartite graphs

(Sum Constraint)
Build your own cubic problem!

**Step 1:** Take the basis problem

- \( \Theta(n^3) \) \( O(n^\omega) \)

(Triangle Detection)

**Step 2:** Choose your toppings

- \( \Theta(n^3) \) \( O(n^3 - \epsilon) \)

(Equal Constraint)

we assume tripartite graphs

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

(Sum Constraint)
Build your own cubic problem!

**Step 1:** Take the basis problem

\[ \Theta(n^3) \ O(n^\omega) \]

(Triangle Detection)

we assume tripartite graphs

\( \sum_{e} \ |w(e)| \leq O(n^2) \)

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

**Step 2:** Choose your toppings

\[ \Theta(n^3) \ O(n^3-\varepsilon) \]

(Equal Constraint)

(Sum Constraint)
Build your own cubic problem!

**Step 1:** Take the basis problem

\[ \Theta(n^3) \quad O(n^\omega) \]

(Triangle Detection)

we assume tripartite graphs

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

“Constrained Triangle Detection”

\[ \sum_e |w(e)| \leq O(n^2) \]

**Step 2:** Choose your toppings

\[ \Theta(n^3) \quad O(n^3 - \epsilon) \]

(Equal Constraint)

(Sum Constraint)
Algorithms for Properties of Hardness $h \leq 2$
$k = 3$: Reduction to Constrained Triangles

$2 < h < k$

$h = k$

$2 = h < k$

$h \leq 1$
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

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

Think of $x_1, x_2, x_3$ as vectors.
$k = 3$: Reduction to Constrained Triangles

Think of $x_1, x_2, x_3$ as vectors.

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

$2 < h < k$

$h = k$

$2 = h < k$

$h \leq 1$
Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

think of $x_1, x_2, x_3$ as vectors

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

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

$2 < h < k$

$h = k$

$2 = h < k$

$h \leq 1$
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

$h = \frac{2}{3}$

$2 < h < 3$

$h = k$

$2 = h < k$

$h \leq 1$
Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

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

Think of $x_1, x_2, x_3$ as vectors.

Insert all edges.

$O(m)$ vertices

$2 < h < k$

$h = k$

$2 = h < k$

$h \leq 1$
Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

Think of $x_1, x_2, x_3$ as vectors

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

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

Insert all edges

$O(m)$ vertices

$2 < h < k$

$h = k$

$2 = h < k$

$h \leq 1$
Algorithms for Properties of Hardness $h \leq 2$

**$k = 3$: Reduction to Constrained Triangles**

Think of $x_1, x_2, x_3$ as vectors.

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

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

Insert all edges.

$O(m)$ vertices.

**$k > 3$: 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 $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

Think of $x_1, x_2, x_3$ as vectors

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

Insert all edges

O(m) vertices

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

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

$$\exists x_1 \exists x_2 \exists x_3 \exists x_4 \forall i: \varphi(0/1, x_2[i], x_3[i], x_4[i])$$
Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

Think of $x_1, x_2, x_3$ as vectors

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

Insert all edges

$O(m)$ vertices

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

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

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

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

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

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

insert all edges

$O(m)$ vertices

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

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

Idea: repeat the above reduction and combine the triangle constraints

Examples

- $2 < h < k$
- $2 = h < k$
- $h \leq 1$
- $h = k$

$h$ vs. $k$ graph with examples for $h$ and $k$ values.
Algorithms for Properties of Hardness $h \leq 2$

$k = 3$: Reduction to Constrained Triangles

- Think of $x_1, x_2, x_3$ as vectors
- Insert all edges
- Idea: spend $O(m^2)$ time to encode $\varphi$ by Equal and Sum constraints

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

- $\exists x_1 \exists x_2 \exists x_3 \forall i: \varphi(x_1[i], x_2[i], x_3[i])$
- $\exists x_1 \exists x_2 \exists x_3 \exists x_4 \forall i: \varphi(0/1, x_2[i], x_3[i], x_4[i])$
- Idea: repeat the above reduction and combine the triangle constraints

Examples

- $\exists x_1 \exists x_2 \exists x_3 \forall i: \text{NAE}(x_1[i], x_2[i], x_3[i])$
**Algorithms for Properties of Hardness $h \leq 2$**

### $k = 3$: Reduction to Constrained Triangles

- **Think of** $x_1, x_2, x_3$ as vectors
- **Idea:** spend $O(m^2)$ time to encode $\varphi$ by Equal and Sum constraints
- Insert all edges
- **O(m) vertices**

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

- **Think of** $x_1, x_2, x_3, x_4$ as vectors
- **Idea:** repeat the above reduction and combine the triangle constraints

### Examples

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

- **$\exists x_1 \exists x_2 \exists x_3 \forall i$:**
  
  $\varphi(0/1, x_2[i], x_3[i], x_4[i])$

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

- $2 < h < k$
- $2 = h < k$
- $h \leq 1$
- $h = k$
Algorithms for Properties of Hardness \( h \leq 2 \)

**\( k = 3 \): Reduction to Constrained Triangles**

Think of \( x_1, x_2, x_3 \) as vectors

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

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

Insert all edges

\( O(m) \) vertices

**\( k > 3 \): Brute-force \( k - 3 \) quantifiers**

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

Idea: repeat the above reduction and combine the triangle constraints

Examples

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

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

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

\[ \exists x_1 \, \exists x_2 \, \exists x_3 \, \forall i: \]
\[ x_1[i] + x_2[i] + x_3[i] \neq 2 \]
How to Employ Sum and Equal Constraints

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

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

\[ \forall i: \text{NAE}(x_1[i], x_2[i], x_3[i]) \]

\[ x_1[i] + x_2[i] + x_3[i] \neq 2 \]

Idea: Exclude 000 and 111 by a Sum constraint

Exclude 111 and 000 in all dimensions

\[ 0 = \| x_1 \land x_2 \land x_3 \|_1 = \| \bar{x}_1 \land \bar{x}_2 \land \bar{x}_3 \|_1 \]
How to Employ Sum and Equal Constraints

∃x₁, ∃x₂, ∃x₃ ∀i: NAE(x₁[i], x₂[i], x₃[i])

Idea: Exclude 000 and 111 by a Sum constraint

Exclude 111 and 000 in all dimensions

⇒ 0 = ||x₁ ∧ x₂ ∧ x₃||₁ = ||x₁ ∧ x₂ ∧ x₃||₁

⇒ 0 = ||x₁ ∧ x₂ ∧ x₃||₁ + ||x₁ ∧ x₂ ∧ x₃||₁

∃x₁, ∃x₂, ∃x₃ ∀i: x₁[i] + x₂[i] + x₃[i] ≠ 2
How to Employ Sum and Equal Constraints

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

**Idea:** Exclude 000 and 111 by a Sum constraint

Exclude 111 and 000 in all dimensions

\[ \Leftrightarrow 0 = \|x_1 \land x_2 \land x_3\|_1 = \|\overline{x}_1 \land \overline{x}_2 \land \overline{x}_3\|_1 \]
\[ \Leftrightarrow 0 = \|x_1 \land x_2 \land x_3\|_1 + \|\overline{x}_1 \land \overline{x}_2 \land \overline{x}_3\|_1 \]
\[ \Leftrightarrow 0 = \|x_1 \land x_2 \land x_3\|_1 - \|x_1 \land x_2 \land x_3\|_1 \]
\[ + \|x_1 \land x_2\|_1 - \|x_1\|_1 \]
\[ + \|x_2 \land x_3\|_1 - \|x_2\|_1 \]
\[ + \|x_3 \land x_1\|_1 - \|x_3\|_1 \]
\[ + d \]

\[ \exists x_1 \exists x_2 \exists x_3 \ \forall i: \ x_1[i] + x_2[i] + x_3[i] \neq 2 \]
How to Employ Sum and Equal Constraints

∃x_1 ∃x_2 ∃x_3 ∀i: 

\( \text{NAE}(x_1[i], x_2[i], x_3[i]) \)

Idea: Exclude 000 and 111 by a Sum constraint

Exclude 111 and 000 in all dimensions

\[ 0 = \| x_1 \land x_2 \land x_3 \|_1 = \| \overline{x}_1 \land \overline{x}_2 \land \overline{x}_3 \|_1 \]

\[ 0 = \| x_1 \land x_2 \land x_3 \|_1 + \| \overline{x}_1 \land \overline{x}_2 \land \overline{x}_3 \|_1 \]

\[ 0 = \| x_1 \land x_2 \land x_3 \|_1 - \| x_1 \land x_2 \land x_3 \|_1 \]

+ \( \| x_1 \land x_2 \|_1 - \| x_1 \|_1 \)

+ \( \| x_2 \land x_3 \|_1 - \| x_2 \|_1 \)

+ \( \| x_3 \land x_1 \|_1 - \| x_3 \|_1 \)

+ d

\[ \exists x_1 \exists x_2 \exists x_3 \forall i: x_1[i] + x_2[i] + x_3[i] \neq 2 \]
How to Employ Sum and Equal Constraints

Exclude 111 and 000 in all dimensions

\[
\Leftrightarrow 0 = \|x_1 \land x_2 \land x_3\|_1 = \|\overline{x}_1 \land \overline{x}_2 \land \overline{x}_3\|_1
\]

\[
\Leftrightarrow 0 = \|x_1 \land x_2 \land x_3\|_1 + \|\overline{x}_1 \land \overline{x}_2 \land \overline{x}_3\|_1
\]

\[
\Leftrightarrow 0 = \|x_1 \land x_2 \land x_3\|_1 - \|x_1 \land x_2 \land x_3\|_1
\]

\[
+ \|x_1 \land x_2\|_1 - \|x_1\|_1 \quad w(x_1, x_2) \text{ cancels!}
\]

\[
+ \|x_2 \land x_3\|_1 - \|x_2\|_1 \quad w(x_2, x_3)
\]

\[
+ \|x_3 \land x_1\|_1 - \|x_3\|_1 \quad w(x_3, x_1)
\]

\[
+ d \quad \text{target t}
\]
How to Employ Sum and Equal Constraints

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

Exclude 111 and 000 in all dimensions

\[ 0 = ||x_1 \land x_2 \land x_3||_1 = ||\overline{x}_1 \land \overline{x}_2 \land \overline{x}_3||_1 \]

\[ 0 = ||x_1 \land x_2 \land x_3||_1 + ||\overline{x}_1 \land \overline{x}_2 \land \overline{x}_3||_1 \]

\[ 0 = \left[ ||x_1 \land x_2 \land x_3||_1 - ||x_1 \land x_2 \land x_3||_1 \right] + ||x_1 \land x_2||_1 - ||x_1||_1 w(x_1, x_2) \]

\[ + ||x_2 \land x_3||_1 - ||x_2||_1 w(x_2, x_3) \]

\[ + ||x_3 \land x_1||_1 - ||x_3||_1 w(x_3, x_1) \]

\[ + d \text{ target t} \]

Generalizes for any pair of falsifying assignments of odd Hamming distance
How to Employ Sum and Equal Constraints

Exclude 111 and 000 in all dimensions

\[ 0 = \|x_1 \land x_2 \land x_3\|_1 = \|\overline{x}_1 \land \overline{x}_2 \land \overline{x}_3\|_1 \]

\[ 0 = \|x_1 \land x_2 \land x_3\|_1 + \|\overline{x}_1 \land \overline{x}_2 \land \overline{x}_3\|_1 \]

\[ 0 = \|x_1 \land x_2 \land x_3\|_1 - \|x_1 \land x_2 \land x_3\|_1 \]

\[ + \|x_1 \land x_2\|_1 - \|x_1\|_1 \]

\[ + \|x_2 \land x_3\|_1 - \|x_2\|_1 \]

\[ + \|x_3 \land x_1\|_1 - \|x_3\|_1 \]

\[ + d \]

Generalizes for any pair of falsifying assignments of odd Hamming distance

Idea: Exclude 000 and 111 by a Sum constraint

Idea: Exclude 110 and 101 by an Equal constraint
How to Employ Sum and Equal Constraints

Exclude 111 and 000 in all dimensions

\[ 0 = \| x_1 \land x_2 \land x_3 \|_1 = \| \bar{x}_1 \land \bar{x}_2 \land \bar{x}_3 \|_1 \]
\[ 0 = \| x_1 \land x_2 \land x_3 \|_1 = \| x_1 \land x_2 \land x_3 \|_1 + \| \bar{x}_1 \land \bar{x}_2 \land \bar{x}_3 \|_1 \]
\[ 0 = \| x_1 \land x_2 \land x_3 \|_1 - \| x_1 \land x_2 \land x_3 \|_1 + \| x_1 \land x_2 \|_1 - \| x_1 \|_1 w(x_1, x_2) + \| x_2 \land x_3 \|_1 - \| x_2 \|_1 w(x_2, x_3) + \| x_3 \land x_1 \|_1 - \| x_3 \|_1 w(x_3, x_1) + d_{\text{target } t} \]

Idea: Exclude 000 and 111 by a Sum constraint

Exclude 110 and 101 in all dimensions

\[ x_1[i] + x_2[i] + x_3[i] \neq 2 \]

Idea: Exclude 110 and 101 by an Equal constraint

Exclude 110 and 101 in all dimensions

\[ x_1 \land x_2 = x_1 \land x_3 \]

Generalizes for any pair of falsifying assignments of odd Hamming distance
How to Employ Sum and Equal Constraints

Exclude 111 and 000 in all dimensions
\[ 0 = \| x_1 \land x_2 \land x_3 \|_1 = \| \bar{x}_1 \land \bar{x}_2 \land \bar{x}_3 \|_1 \]
\[ 0 = \| x_1 \land x_2 \land x_3 \|_1 + \| \bar{x}_1 \land \bar{x}_2 \land \bar{x}_3 \|_1 \]
\[ 0 = \| x_1 \land x_2 \land x_3 \|_1 - \| x_1 \land x_2 \land x_3 \|_1 - \| x_1 \|_1 - \| x_2 \|_1 - \| x_3 \|_1 + \text{d} \]

Exclude 110 and 101 in all dimensions
\[ x_1[i] + x_2[i] + x_3[i] \neq 2 \]
\[ x_1[i] + x_2[i] = x_1[i] + x_3[i] \]

Generalizes for any pair of falsifying assignments of odd Hamming distance
How to Employ Sum and Equal Constraints

Exclude 111 and 000 in all dimensions

\[ 0 = \| X_1 \land X_2 \land X_3 \|_1 = \| \overline{X}_1 \land \overline{X}_2 \land \overline{X}_3 \|_1 \]

\[ 0 = \| X_1 \land X_2 \land X_3 \|_1 + \| \overline{X}_1 \land \overline{X}_2 \land \overline{X}_3 \|_1 \]

\[ 0 = \| X_1 \land X_2 \land X_3 \|_1 - \| X_1 \land X_2 \land X_3 \|_1 \]

+ \| X_1 \land X_2 \|_1 - \| X_1 \|_1 \]

+ \| X_2 \land X_3 \|_1 - \| X_2 \|_1 \]

+ \| X_3 \land X_1 \|_1 - \| X_3 \|_1 \]

+ \| X_1 \land X_2 \|_1 - \| X_1 \|_1 \]

\[ w(x_1, x_2) \]

\[ w(x_1, x_3) \]

Generalizes for any pair of falsifying assignments of odd Hamming distance

Exclude 110 and 101 in all dimensions

\[ \exists X_1 \ \exists X_2 \ \exists X_3 \ \forall i: W(x_1[i], x_2[i], x_3[i]) \]

\[ \exists X_1 \ \exists X_2 \ \exists X_3 \ \forall i: X[i]_1 + X[i]_2 + X[i]_3 \neq 2 \]

Idea: Exclude 110 and 101 by an Equal constraint

Idea: Exclude 000 and 111 by a Sum constraint

\[ \text{convert vectors arbitrarily into numbers} \]
How to Employ Sum and Equal Constraints

Idea: Exclude 000 and 111 by a Sum constraint

Exclude 111 and 000 in all dimensions
\[
0 = \|x_1 \land x_2 \land x_3 \|_1 = \|\overline{x}_1 \land \overline{x}_2 \land \overline{x}_3 \|_1
\]
\[
0 = \|x_1 \land x_2 \land x_3 \|_1 + \|\overline{x}_1 \land \overline{x}_2 \land \overline{x}_3 \|_1
\]
\[
0 = \|x_1 \land x_2 \land x_3 \|_1 - \|x_1 \land x_2 \land x_3 \|_1 - \|x_1 \|_1 - \|x_2 \|_1 - \|x_3 \|_1 + d_{\text{target } t}
\]

cancels!

Generalizes for any pair of falsifying assignments of odd Hamming distance

Idea: Exclude 110 and 101 by an Equal constraint

Exclude 110 and 101 in all dimensions
\[
\iff \ x_1 \land x_2 = x_1 \land x_3
\]
\[
\iff \ w(x_1, x_2) = w(x_1, x_3)
\]

convert vectors arbitrarily into numbers

Generalizes for any pair of falsifying assignments of even Hamming distance
Conclusion and Open Problems

- 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 $h$?

the same dichotomy holds in the counting setting