

Simulating Logspace-Recursion with Logarithmic Quantifier Depth

38th Symposium on Logic in Computer Science

Steffen van Bergerem

Martin Grohe

Sandra Kiefer

Luca Oeljeklaus

June 26th, 2023

The Weisfeiler-Leman Algorithm (WL_k): a subroutine for GI-testing

- A subroutine for Graph Isomorphism testing

The Weisfeiler-Leman Algorithm (WL_k): a subroutine for GI-testing

- A subroutine for Graph Isomorphism testing
- Iteratively refines a partition of vertex k -tuples

The Weisfeiler-Leman Algorithm (WL_k): a subroutine for GI-testing

- A subroutine for **Graph Isomorphism** testing
- **Iteratively** refines a **partition of vertex k -tuples**
- For fixed k , WL_k has a polynomial time implementation¹

¹*Immerman, Lander 1990*

The Weisfeiler-Leman Algorithm (WL_k): a subroutine for GI-testing

- A subroutine for **Graph Isomorphism** testing
- **Iteratively** refines a **partition of vertex k -tuples**
- For fixed k , WL_k has a polynomial time implementation¹
- **Identifies** a graph if it **distinguishes it** from any non-isomorphic one

¹*Immerman, Lander 1990*

The Weisfeiler-Leman Algorithm (WL_k): a subroutine for GI-testing

- A subroutine for **Graph Isomorphism** testing
- **Iteratively** refines a **partition of vertex k -tuples**
- For fixed k , WL_k has a polynomial time implementation¹
- **Identifies** a graph if it **distinguishes it** from any non-isomorphic one
- **WL-dimension** of a graph: the smallest k such that WL_k identifies it

¹*Immerman, Lander 1990*

The Weisfeiler-Leman Algorithm (WL_k): a subroutine for GI-testing

- A subroutine for **Graph Isomorphism** testing
- **Iteratively** refines a **partition of vertex k -tuples**
- For fixed k , WL_k has a polynomial time implementation¹
- **Identifies** a graph if it **distinguishes it** from any non-isomorphic one
- **WL-dimension** of a graph: the smallest k such that WL_k identifies it
- **Upper bounds** have been found for many graph classes, including:
 - Planar² Bounded Treewidth³ Bounded Rankwidth⁴
 - Interval⁵ Bounded Genus⁶ Excluded Minors⁷

¹Immerman, Lander 1990 ²Kiefer et al. 2019 ³Grohe, Mariño 1999 ⁴ Grohe, Neuen 2019

⁵Evdokimov et al. 2000 ⁶Grohe, Kiefer 2019 ⁷Grohe 2012

- The **number of iterations** of WL_k measures the parallelisability of the algorithm

Logarithmic Weisfeiler-Leman ($WL_k^{\mathcal{O}(\log n)}$)

- The **number of iterations** of WL_k measures the parallelisability of the algorithm
- A **logarithmic bound** for isomorphism puts the problem in TC^1 ¹

¹*Grohe, Verbitsky 2006*

Logarithmic Weisfeiler-Leman ($WL_k^{\mathcal{O}(\log n)}$)

- The **number of iterations** of WL_k measures the parallelisability of the algorithm
- A **logarithmic bound** for isomorphism puts the problem in TC^1 ¹
- This bound suffices for
Embeddable Graphs¹ Bounded Treewidth Graphs¹ Planar Graphs²

¹Grohe, Verbitsky 2006 ²Grohe, Kiefer 2021

Logarithmic Weisfeiler-Leman ($WL_k^{\mathcal{O}(\log n)}$)

- The **number of iterations** of WL_k measures the parallelisability of the algorithm
- A **logarithmic bound** for isomorphism puts the problem in TC^1 ¹
- This bound suffices for
 - Embeddable Graphs¹
 - Bounded Treewidth Graphs¹
 - Planar Graphs²
 - Interval Graphs³
 - Claw-free chordal graphs³

¹Grohe, Verbitsky 2006 ²Grohe, Kiefer 2021 ³van Bergerem et al. 2023

$WL_k^{O(\log n)}$ is as expressive as log-depth $(k + 1)$ -variable counting-FO

C_k^l :

$$\underbrace{\exists^{\geq 2} x. \left[\underbrace{\forall y. (E(xy) \wedge \underbrace{\forall x. x = x})}_{\text{blue}} \right]}_{\text{red}} \in C_2^3$$

$WL_k^{O(\log n)}$ is as expressive as log-depth $(k + 1)$ -variable counting-FO

C_k^l :

- Extend FO by counting quantifiers

$$\underbrace{\exists^{\geq 2} x. \left[\underbrace{\forall y. (E(xy) \wedge \underbrace{\forall x. x = x})}_{\text{true}} \right]}_{\text{true}} \in C_2^3$$

$WL_k^{O(\log n)}$ is as expressive as log-depth $(k + 1)$ -variable counting-FO

C_k^l :

- Extend FO by counting quantifiers
- Restrict to k variables

$$\underbrace{\exists^{\geq 2} x. \left[\underbrace{\forall y. (E(xy) \wedge \underbrace{\forall x. x = x})}_{\text{blue}} \right]}_{\text{red}} \in C_2^3$$

$WL_k^{O(\log n)}$ is as expressive as log-depth $(k + 1)$ -variable counting-FO

C_k^ℓ :

- Extend FO by counting quantifiers
- Restrict to k variables
- Allow only quantifier depth $\leq \ell$

$$\underbrace{\exists^{\geq 2} x. \left[\underbrace{\forall y. (E(xy) \wedge \underbrace{\forall x. x = x})}_{\text{blue}} \right]}_{\text{red}} \in C_2^3$$

$WL_k^{\mathcal{O}(\log n)}$ is as expressive as log-depth $(k + 1)$ -variable counting-FO

C_k^ℓ :

- Extend FO by counting quantifiers
- Restrict to k variables
- Allow only quantifier depth $\leq \ell$

$$\underbrace{\exists^{\geq 2} x. \left[\underbrace{\forall y. (E(xy) \wedge \underbrace{\forall x. x = x})}_{\text{trivial}} \right]}_{\text{counting quantifier}} \in C_2^3$$

A property P can be expressed in $C_k^{\mathcal{O}(\log n)}$ if there is an $f(n) \in \mathcal{O}(\log n)$ such that, for all $n \in \mathbb{N}$, there is an $C_k^{f(n)}$ -formula which defines P on structures of order n .

$WL_k^{\mathcal{O}(\log n)}$ is as expressive as log-depth $(k + 1)$ -variable counting-FO

C_k^ℓ :

- Extend FO by counting quantifiers
- Restrict to k variables
- Allow only quantifier depth $\leq \ell$

$$\underbrace{\exists^{\geq 2} x. \left[\underbrace{\forall y. (E(xy) \wedge \underbrace{\forall x. x = x})}_{\text{trivial}} \right]}_{\text{counting quantifier}} \in C_2^3$$

A property P can be expressed in $C_k^{\mathcal{O}(\log n)}$ if there is an $f(n) \in \mathcal{O}(\log n)$ such that, for all $n \in \mathbb{N}$, there is an $C_k^{f(n)}$ -formula which defines P on structures of order n .

Lemma. (Cai, Fürer, Immerman 1992)

Let $k, \ell \in \mathbb{N}_{>0}$. WL_k^ℓ distinguishes two graphs G and H iff they disagree on some C_{k+1}^ℓ -sentence φ .

Goal: $C_k^{\mathcal{O}(\log n)}$ identifies all interval graphs.

Goal: $C_k^{O(\log n)}$ identifies all interval graphs.

Known Result: $LREC_=$ identifies all interval graphs.

Goal: $C_k^{\mathcal{O}(\log n)}$ identifies all interval graphs.

Known Result: $LREC_=$ identifies all interval graphs.

Key Result: $C_k^{\mathcal{O}(\log n)}$ “captures” $LREC_=$ (non-uniformly).

LREC₌: An attempt to capture LOGSPACE

- An extension of FO+C by a **limited recursion operator**¹

¹*Grohe et al. 2011*

LREC₌: An attempt to capture LOGSPACE

- An extension of FO+C by a **limited recursion operator**¹
- Captures LOGSPACE on directed trees, interval graphs, claw-free chordal graphs^{1,2}

¹*Grohe et al. 2011* ²*Grußien 2017*

LREC₌: An attempt to capture LOGSPACE

- An extension of FO+C by a **limited recursion operator**¹
- Captures LOGSPACE on directed trees, interval graphs, claw-free chordal graphs^{1,2}
- $\text{STC+C} \preceq \text{LREC}_=$

¹Grohe et al. 2011 ²Grußien 2017

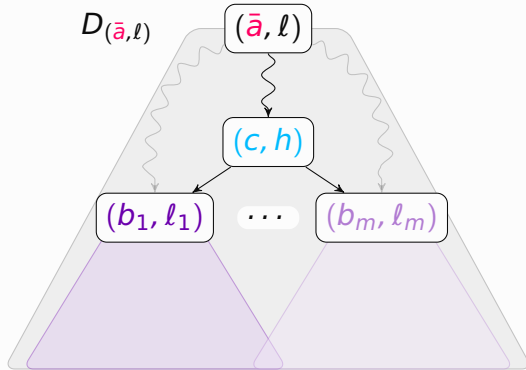
LREC₌: An attempt to capture LOGSPACE

- An extension of FO+C by a **limited recursion operator**¹
- Captures LOGSPACE on directed trees, interval graphs, claw-free chordal graphs^{1,2}
- $STC+C \preceq LREC_{=} \preceq \overset{FP+C}{CLogspace} \preceq LOGSPACE$ ^{1,3}

¹Grohe et al. 2011 ²Grußien 2017 ³Grädel, Schalthöfer 2019

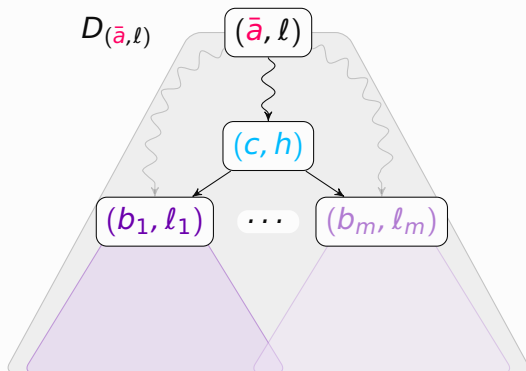
The computation can be described as a logarithmic-depth tree

$$[\text{Irec}_{\bar{x}, \bar{y}, t \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, \ell)$$



The computation can be described as a logarithmic-depth tree

$$[\text{Irec}_{\bar{x}, \bar{y}, l \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, l)$$

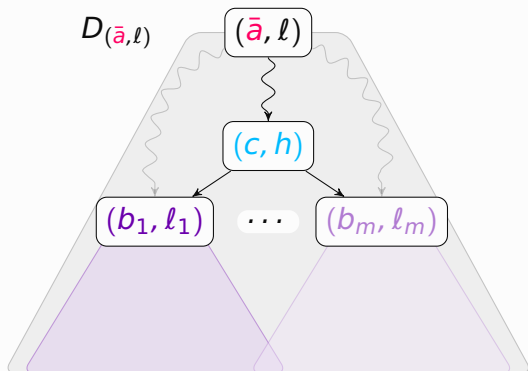


For a DAG rooted in a node (\bar{a}, l) , there exists a node (c, h) such that:

- Every $(b_1, l_1) \dots (b_m, l_m)$ requires at most half as much computation as (\bar{a}, l) .

The computation can be described as a logarithmic-depth tree

$$[\text{Irec}_{\bar{x}, \bar{y}, l \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, l)$$

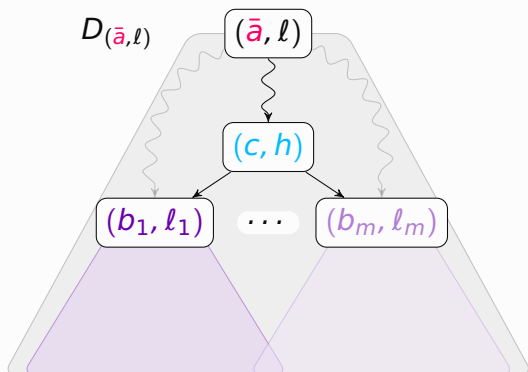


For a DAG rooted in a node (\bar{a}, l) , there exists a node (c, h) such that:

- Every $(b_1, l_1) \dots (b_m, l_m)$ requires at most half as much computation as (\bar{a}, l) .
- A similar statement holds for the remaining computation.

The computation can be described as a logarithmic-depth tree

$$[\text{Irec}_{\bar{x}, \bar{y}, t \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, \ell)$$



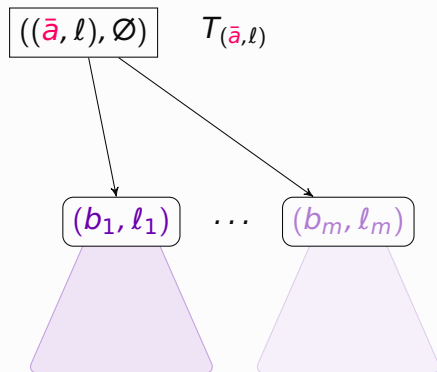
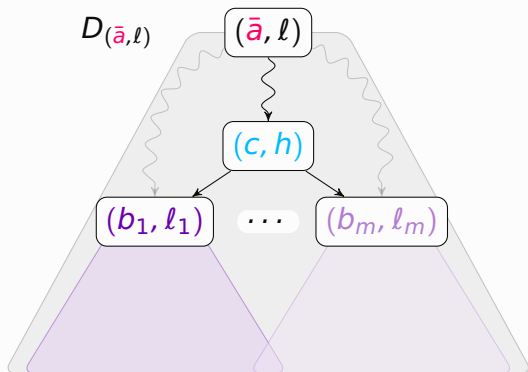
$$((\bar{a}, \ell), \emptyset) \quad T(\bar{a}, \ell)$$

For a DAG rooted in a node (\bar{a}, ℓ) , there exists a node (c, h) such that:

- Every $(b_1, l_1) \dots (b_m, l_m)$ requires at most half as much computation as (\bar{a}, ℓ) .
- A similar statement holds for the remaining computation.

The computation can be described as a logarithmic-depth tree

$$[\text{Irec}_{\bar{x}, \bar{y}, l \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, l)$$

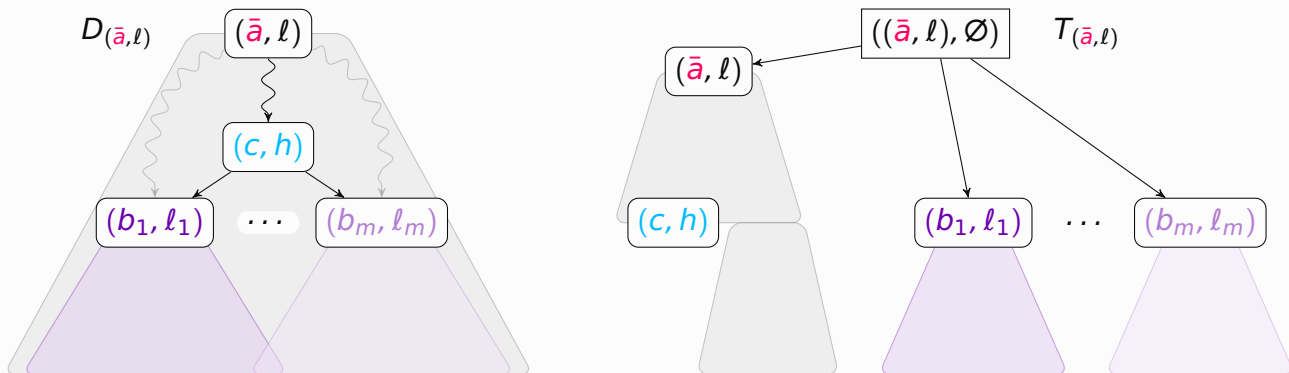


For a DAG rooted in a node (\bar{a}, l) , there exists a node (c, h) such that:

- Every $(b_1, l_1) \dots (b_m, l_m)$ requires at most half as much computation as (\bar{a}, l) .
- A similar statement holds for the remaining computation.

The computation can be described as a logarithmic-depth tree

$$[\text{Irec}_{\bar{x}, \bar{y}, l \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, l)$$



For a DAG rooted in a node (\bar{a}, l) , there exists a node (c, h) such that:

- Every $(b_1, l_1) \dots (b_m, l_m)$ requires at most half as much computation as (\bar{a}, l) .
- A similar statement holds for the remaining computation.

Using families of $C_k^{\mathcal{O}(\log n)}$ -formulae, check:

- The existence of such a treelike decomposition, and

Using families of $C_k^{\mathcal{O}(\log n)}$ -formulae, check:

- The existence of such a treelike decomposition, and
- The correctness of the computation.

Using families of $C_k^{\mathcal{O}(\log n)}$ -formulae, check:

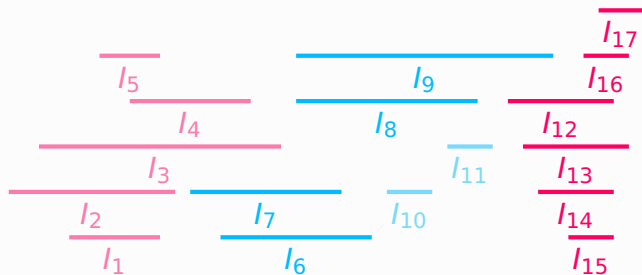
- The existence of such a treelike decomposition, and
- The correctness of the computation.

Theorem. $\text{LREC}_= C_k^{\mathcal{O}(\log n)}$ -simulability

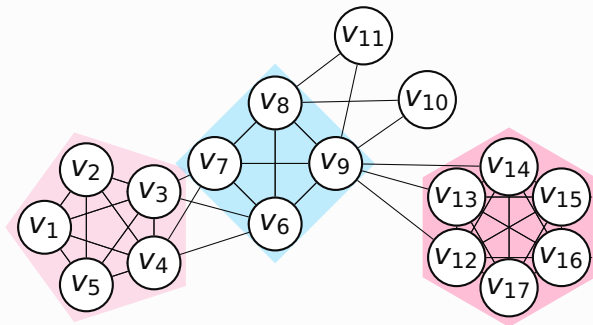
For every $\text{LREC}_=$ -formula $\psi(\bar{x}, \bar{\mu})$ there exists a k such that, for all n -vertex graphs G , there is an equivalent family $C_k^{\mathcal{O}(\log n)}$ -formulae $(\varphi_{\bar{m}}(\bar{x}))$.

Application to Interval Graphs

Interval Graphs: Connect overlapping intervals by edges



A set of intervals.



The corresponding interval graph.

Interval graphs have applications in e.g. operations research and biology.

Lemma. (Grohe et al. 2011)

LREC₌ identifies all interval graphs.

Lemma. (Grohe et al. 2011)

$LREC_=$ identifies all interval graphs.

Theorem. $LREC_= C_k^{\mathcal{O}(\log n)}$ -simulability

For every $LREC_=$ -formula $\psi(\bar{x}, \bar{\mu})$ there exists a k such that, for all n -vertex graphs G , there is an equivalent family $C_k^{\mathcal{O}(\log n)}$ -formulae $(\varphi_{\bar{m}}(\bar{x}))$.

Lemma. (Grohe et al. 2011)

$LREC_=$ identifies all interval graphs.

Theorem. $LREC_= C_k^{\mathcal{O}(\log n)}$ -simulability

For every $LREC_=$ -formula $\psi(\bar{x}, \bar{\mu})$ there exists a k such that, for all n -vertex graphs G , there is an equivalent family $C_k^{\mathcal{O}(\log n)}$ -formulae $(\varphi_{\bar{m}}(\bar{x}))$.

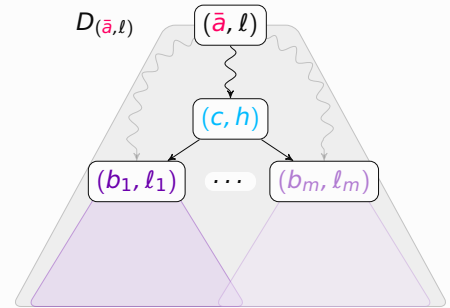
Theorem. $C_k^{\mathcal{O}(\log n)}$ on interval graphs

There is a k such that every n -vertex interval graph is identified by $C_k^{\mathcal{O}(\log n)}$.

Conclusion

What we have discussed:

$$[\text{Irec}_{\bar{x}, \bar{y}, i \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, l)$$



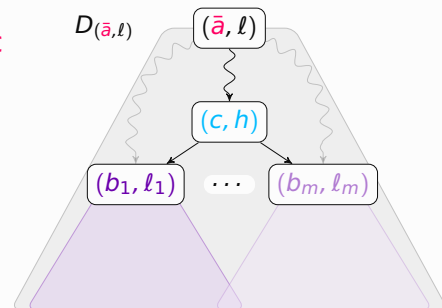
$$\Phi_G := \left(\bigwedge_{i, j \in [n]} \varphi_{ij} \leftrightarrow \varphi_{ij}^G \right) \wedge \varphi_{\text{interval}}^{(n)}$$

Conclusion

What we have discussed:

- $\text{LREC}_=$ and how to simulate its operator using logarithmic quantifier depth C_k .

$$[\text{lrec}_{\bar{x}, \bar{y}, i \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, l)$$



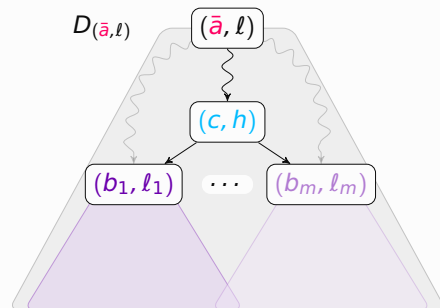
$$\Phi_G := \left(\bigwedge_{i, j \in [n]} \varphi_{ij} \leftrightarrow \varphi_{ij}^G \right) \wedge \varphi_{\text{interval}}^{(n)}$$

Conclusion

What we have discussed:

- $\text{LREC}_=$ and how to **simulate** its **operator** using **logarithmic quantifier depth** C_k .
- Applying this to an $\text{LREC}_=$ -definable canonisation yields that **log-WL identifies all interval graphs**.

$$[\text{lrec}_{\bar{x}, \bar{y}, i \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, l)$$



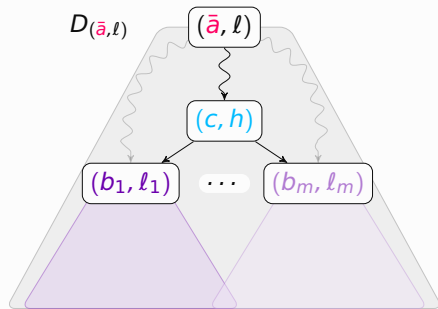
$$\Phi_G := \left(\bigwedge_{i, j \in [n]} \varphi_{ij} \leftrightarrow \varphi_{ij}^G \right) \wedge \varphi_{\text{interval}}^{(n)}$$

Conclusion

What we have discussed:

- $\text{LREC}_=$ and how to **simulate** its **operator** using **logarithmic quantifier depth** C_k .
- Applying this to an $\text{LREC}_=$ -definable canonisation yields that **log-WL identifies all interval graphs**.
- A similar result can be obtained for **claw-free chordal graphs**.

$$[\text{lrec}_{\bar{x}, \bar{y}, i \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, l)$$



$$\Phi_G := \left(\bigwedge_{i, j \in [n]} \varphi_{ij} \leftrightarrow \varphi_{ij}^G \right) \wedge \varphi_{\text{interval}}^{(n)}$$

Conclusion

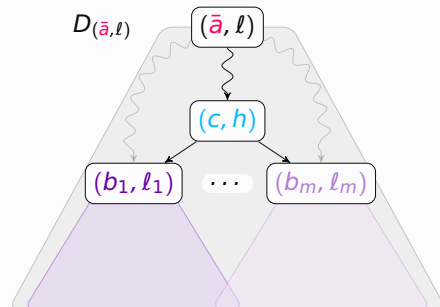
What we have discussed:

- $\text{LREC}_=$ and how to **simulate** its **operator** using **logarithmic quantifier depth** C_k .
- Applying this to an $\text{LREC}_=$ -definable canonisation yields that **log-WL identifies all interval graphs**.
- A similar result can be obtained for **claw-free chordal graphs**.

Open Questions:

- Further exploration of the **power of log-WL**. Are there **other graph classes** identified by log-WL?

$$[\text{lrec}_{\bar{x}, \bar{y}, i \leq t} \varphi =, \varphi_E, \varphi_C](\bar{a}, l)$$



$$\Phi_G := \left(\bigwedge_{i, j \in [n]} \varphi_{ij} \leftrightarrow \varphi_{ij}^G \right) \wedge \varphi_{\text{interval}}^{(n)}$$