Weisfeiler–Leman Color Refinement, Tree Homomorphisms, and Fractional Isomorphism

Mueed Awais

Euler Circle Seminar
Supervised by Rachana Madhukara (MIT)

July 14, 2025

Outline

Why Weisfeiler–Leman?

- Simple yet powerful graph algorithm: **Color Refinement (1-WL)**.
- Efficient and scalable: O(n log n) rounds, each linear in edges.
- Crucial in:
  - **Graph isomorphism testing**
  - **Graph neural networks (GNNs)**
  - **Logic and descriptive complexity**
- Links combinatorics, algebra, logic, and machine learning.

Historical Origins

- 1968: Weisfeiler and Leman introduce a family of color refinement algorithms.
- Base version: **1-WL**, known earlier in chemistry as the *Morgan algorithm*.
- Used in:
  - Chemical graph comparison
  - Symmetry detection
  - Preprocessing for isomorphism solvers

Historical Origins

- 1968: Weisfeiler and Leman introduce a family of color refinement algorithms.
- Base version: **1-WL**, known earlier in chemistry as the *Morgan algorithm*.
- Used in:
  - Chemical graph comparison
  - Symmetry detection
  - Preprocessing for isomorphism solvers

Cai–Fürer–Immerman (1992): *1-WL* $= C^2$ *logic* $=$ *Tree homomorphism profile*

Color Refinement: Core Idea

- Goal: iteratively improve vertex labels by neighborhood structure.
- Initialize: all nodes given the same color (or degree-based).
- Repeat:
  - Each vertex hashes its current color + multiset of neighbor colors.
  - Reassigns a new color based on that.
- Continue until no change (stable coloring).

Color Refinement: Core Idea

- Goal: iteratively improve vertex labels by neighborhood structure.
- Initialize: all nodes given the same color (or degree-based).
- Repeat:
  - Each vertex hashes its current color + multiset of neighbor colors.
  - Reassigns a new color based on that.
- Continue until no change (stable coloring).

This is the 1-dimensional Weisfeiler–Leman algorithm.

Color Refinement: Formal Definition

Let $G = (V, E)$.

- Initial coloring: $C^{(0)}(v) = 1$ for all $v \in V$.
- At step i:

$$C^{(i+1)}(v) = \text{Hash}\left(C^{(i)}(v),\ \{\{C^{(i)}(u) : u \in N(v)\}\}\right)$$

- Stop when $C^{(i+1)} = C^{(i)}$. The result is $C^{(\infty)}$.

Note: $\{\{\cdot\}\}$ denotes a multiset.

Equitable Partitions

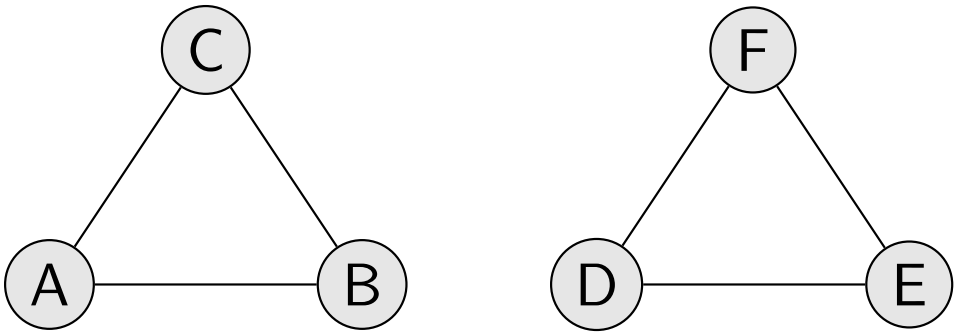
Definition

A vertex coloring is **equitable** if:

$$\forall i, j, \text{ every } v \in C_i \text{ has same number of neighbors in } C_j$$

- WL always converges to an equitable partition.
- Equitable partitions define a **quotient matrix** of the graph.

Color Refinement Example (Graph)



Two triangles: same local structure.

- All vertices have same degree
- 1-WL can't distinguish the components

When 1-WL Fails: Intuition

- WL only sees vertex *types* — based on local neighborhood trees.
- So, graphs with identical multisets of local rooted trees **cannot** be distinguished.
- This includes:
  - Strongly regular graphs
  - Regular disconnected graphs

When 1-WL Fails: Intuition

- WL only sees vertex *types* — based on local neighborhood trees.
- So, graphs with identical multisets of local rooted trees **cannot** be distinguished.
- This includes:
  - Strongly regular graphs
  - Regular disconnected graphs

Key Question: What exactly does 1-WL see — and miss?

Three Equivalent Worlds (Preview)

- 1-WL equivalence = **tree homomorphism equality**
- 1-WL equivalence = **fractional isomorphism**
- Tree hom equality = **fractional isomorphism**

Three Equivalent Worlds (Preview)

- 1-WL equivalence = **tree homomorphism equality**
- 1-WL equivalence = **fractional isomorphism**
- Tree hom equality = **fractional isomorphism**

We will prove: these three characterizations are equivalent.

$$\textit{Combinatorics} \leftrightarrow \textit{Algebra} \leftrightarrow \textit{Enumeration}$$

Next: Graph Homomorphisms

- Next up: definitions and properties of graph homomorphisms.
- Focus: tree homomorphism counts $\hom(T, G)$
- We will show:
  - Tree counts $\Rightarrow$ WL equivalence
  - WL equivalence $\Rightarrow$ fractional isomorphism

Fractional Iso Tree-Hom Equivalence

- Goal: If $A_G X = X A_H$, then $\hom(T, G) = \hom(T, H)$ for all trees T
- We use induction on the number of nodes in the tree
- Fractional matrix X "transports" homomorphisms from G to H

Base Case: Single Vertex Tree

- $T = K_1$ (a single vertex)
- $\hom(K_1, G) = |V(G)| = |V(H)| = \hom(K_1, H)$
- This holds because X is doubly stochastic: it preserves total mass

Inductive Setup

- Let T be a rooted tree with root r
- Children subtrees: $T_1, T_2, \ldots, T_k$
- We will compute $\hom(T, G)$ recursively by:

$$\hom(T, G) = \sum_{v \in V(G)} \prod_{i=1}^{k} \hom(T_i, G \mid r \mapsto v)$$

Conditional Homomorphisms

- Let $\hom(T_i, G \mid r \mapsto u) =$ ways to map T_i to G with root sent to u
- We apply X:

$$\sum_{u \in V(G)} X_{vu} \cdot \hom(T_i, G \mid r \mapsto u) = \hom(T_i, H \mid r \mapsto v)$$

- This lets us recursively carry the homomorphism profile through X

Putting It Together

- Combining subtree counts:

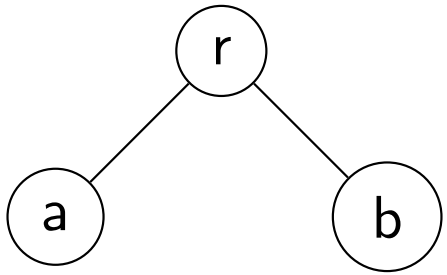
$$\hom(T, G) = \sum_{v \in V(G)} \prod_{i=1}^{k} \hom(T_i, G \mid r \mapsto v)$$

- Apply X entrywise to shift these into H:

$$\hom(T, G) = \hom(T, H)$$

- Thus: fractional isomorphism implies equal tree homomorphism counts

Visual Intuition



A rooted tree T of depth 1. Homomorphisms from T into G can be transferred to H via X

Example

- Let T be a star: central node + 3 leaves
- Homomorphisms into G and H reflect degree profiles
- If X is fractional iso, degrees must align on average
- So the number of star maps is equal in G and H

Proof Summary

- Base case: vertex counts preserved by stochasticity of X
- Inductive step: subtree recursion + X mapping preserves homomorphisms
- Thus: $\hom(T, G) = \hom(T, H)$ for all trees T

Conclusion: Direction 2 Complete

- We've shown:

$$A_G X = X A_H \Rightarrow \hom(T, G) = \hom(T, H)$$

- This bridges linear algebra to enumeration
- Next: show the reverse direction using WL

Goal: Tree Counts WL Equivalence

- Suppose $\hom(T, G) = \hom(T, H)$ for all trees T
- We want to show: $G \sim_{\text{1-WL}} H$
- Strategy: prove the contrapositive
- If 1-WL distinguishes G and H, then $\exists$ tree T with $\hom(T, G) \neq \hom(T, H)$

What Does 1-WL Track?

- It refines vertex colors based on local rooted neighborhoods
- At each round: labels capture tree shapes rooted at each vertex
- Eventually: classifies vertices by tree neighborhoods

What Does 1-WL Track?

- It refines vertex colors based on local rooted neighborhoods
- At each round: labels capture tree shapes rooted at each vertex
- Eventually: classifies vertices by tree neighborhoods

So: different WL colorings $\Rightarrow$ different rooted tree profiles

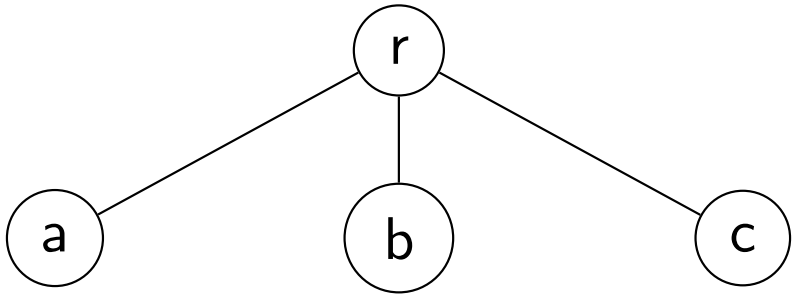
Constructing Distinguishing Tree

- Suppose G and H differ in WL colors at round k
- Let color c appear in G with frequency f and in H with $f' \neq f$
- This means: some rooted tree T explains this difference
- Construct T that reflects the neighborhood profile of c

Color Signatures

- Each vertex at round k has a "signature": multiset of neighbor colors from round $k-1$
- These signatures correspond to rooted trees of depth k
- So a difference in counts of a signature $\Rightarrow$ a difference in homomorphism count for its tree

Example: Color Signature Tree



Tree rooted at r with 3 children — represents a color signature at depth 1

Intuition Recap

- WL distinguishes graphs by "counting" tree signatures
- Each WL color corresponds to a rooted tree type
- Mismatch in WL color count $\Rightarrow$ mismatch in tree homomorphism counts

Formal Statement

Lemma

If $G \not\sim_{1\text{-WL}} H$, then there exists a rooted tree T such that:

$$\hom(T, G) \neq \hom(T, H)$$

Formal Statement

Lemma

If $G \not\sim_{1\text{-WL}} H$, then there exists a rooted tree T such that:

$$\hom(T, G) \neq \hom(T, H)$$

This proves the contrapositive: WL distinguishability tree-count inequality

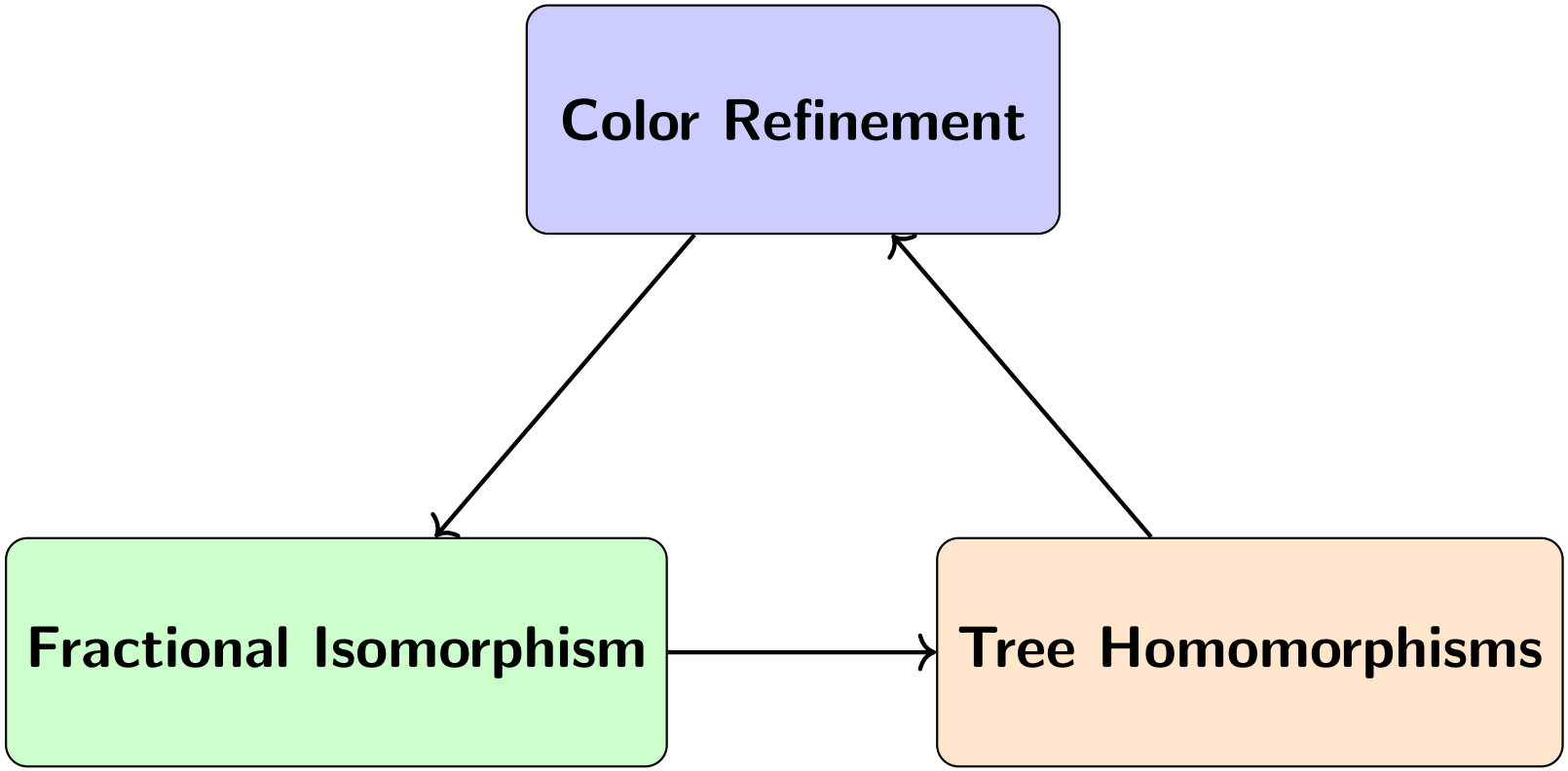
Connecting All Directions

- We've now completed the full cycle:

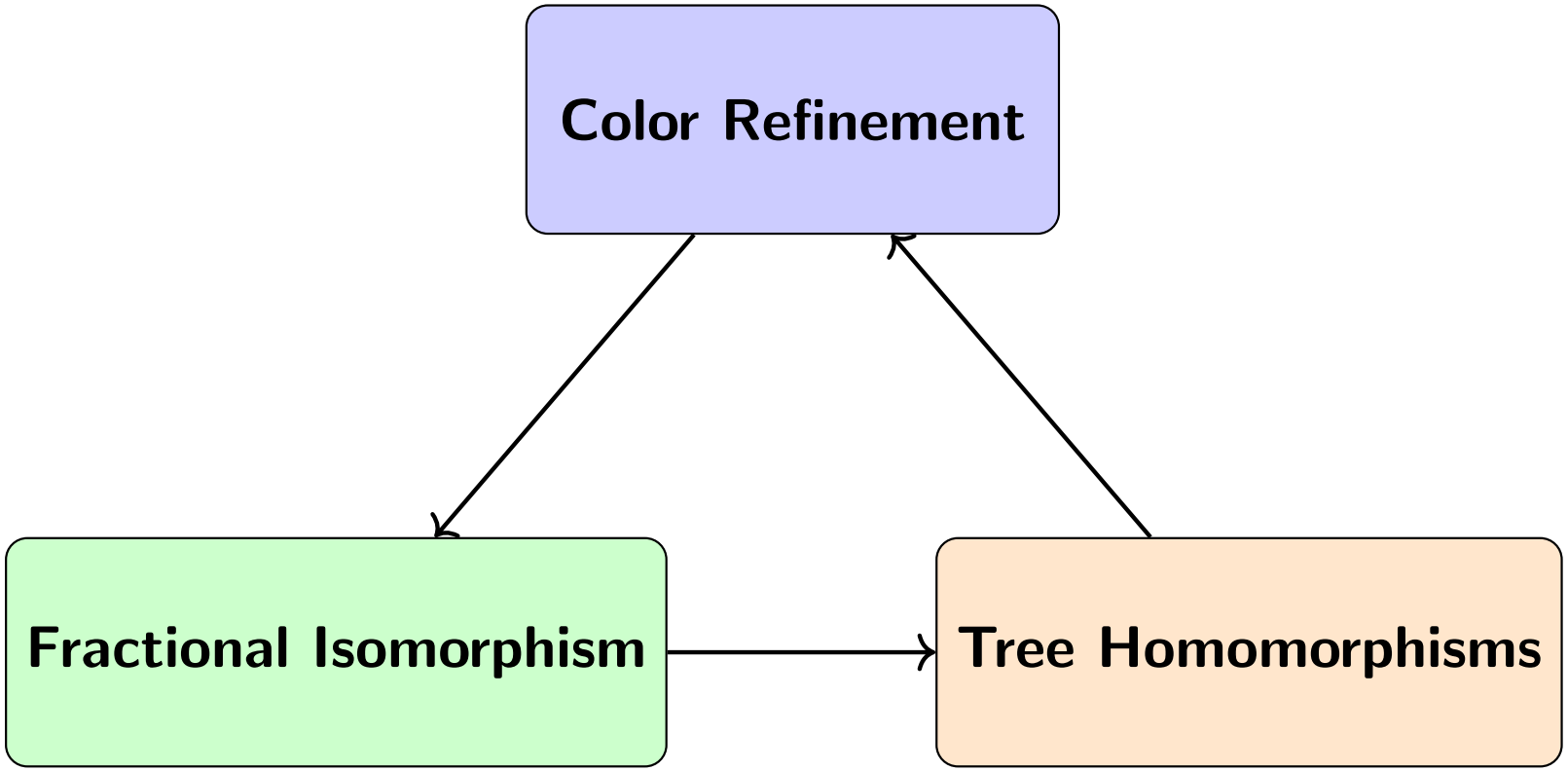
$$\text{WL eq.} \Rightarrow \text{Frac. Iso.} \Rightarrow \text{Tree Hom.} \Rightarrow \text{WL eq.}$$

- Each view strengthens our understanding of graph similarity

Triangle of Equivalences



Triangle of Equivalences



Result:

all three conditions are equivalent!

Proof Summary

- $G \sim_{\text{1-WL}} H$ same WL color profiles
- common equitable partition fractional isomorphism
- tree homomorphism counts agree
- These equivalences provide combinatorial, algebraic, and logical lenses

Applications of 1-WL

- Widely used in:
  - Graph isomorphism testing
  - Graph Neural Networks (GNNs)
  - Graph kernels in ML
- Fast, combinatorial, and scalable
- Central to many modern algorithms despite its simplicity

Example: C_6 vs $2K_3$

- C_6: 6-cycle
- $2K_3$: two disjoint triangles
- Both are 3-regular with 6 vertices
- 1-WL fails to distinguish them
- But: they are not isomorphic

Example: C_6 vs $2K_3$

- C_6: 6-cycle
- $2K_3$: two disjoint triangles
- Both are 3-regular with 6 vertices
- 1-WL fails to distinguish them
- But: they are not isomorphic

This illustrates the limitations of 1-WL!

Example: Shrikhande vs Rook Complement

- Two different strongly regular graphs on 16 vertices
- Have identical WL colorings
- Same spectrum, tree hom counts, fractional isomorphism
- Yet: not isomorphic

Example: Shrikhande vs Rook Complement

- Two different strongly regular graphs on 16 vertices
- Have identical WL colorings
- Same spectrum, tree hom counts, fractional isomorphism
- Yet: not isomorphic

These graphs show deep symmetry and test WL's expressiveness

1-WL and Graph Kernels

- Weisfeiler–Lehman subtree kernel (Shervashidze et al. 2009)
- Uses WL iterations to build label histograms
- Popular in chemical and biological graph classification
- 1-WL forms the backbone of this kernel

1-WL and Graph Neural Networks

- GNNs aggregate neighbor features — just like WL!
- Any standard Message-Passing GNN (MPNN) has power 1-WL
- If 1-WL can't distinguish two graphs, neither can an MPNN
- Limitation $\rightarrow$ motivation for deeper models

Overcoming WL's Limitations

- Use higher-dimensional WL: 2-WL, 3-WL, etc.
- More expressive but more computationally expensive
- In ML: DeepSets, Graph Transformers, or logic-guided GNNs
- Trade-off between expressiveness and efficiency

When Does WL Work Well?

- Almost all random graphs are distinguishable by 1-WL
- It fails mostly on symmetric or regular graphs
- In practice: WL distinguishes most graphs used in applications

When Does WL Work Well?

- Almost all random graphs are distinguishable by 1-WL
- It fails mostly on symmetric or regular graphs
- In practice: WL distinguishes most graphs used in applications

Key insight: it's not perfect, but very powerful in practice

Logic and Descriptive Complexity

- 1-WL C^2 logic (2-variable logic with counting)
- Each WL iteration simulates quantifier patterns in C^2
- Thus: graphs indistinguishable by 1-WL satisfy same C^2 sentences

Logic and Descriptive Complexity

- 1-WL C^2 logic (2-variable logic with counting)
- Each WL iteration simulates quantifier patterns in C^2
- Thus: graphs indistinguishable by 1-WL satisfy same C^2 sentences

Gives a formal logic lens to understand WL's expressiveness

1-WL in Practice

- WL is used in:
  - Chemical informatics (e.g., RDKit)
  - Software decompilation and malware detection
  - Preprocessing for NAUTY / graph isomorphism solvers
- Also: node classification and similarity scoring

Summary: Power and Limits of 1-WL

- Simple, scalable, combinatorial
- Powerful in practice but fails on highly symmetric graphs
- Informs the design of GNNs and graph kernels
- Part of a hierarchy — can be generalized to k-WL

Beyond 1-WL: Higher WL Hierarchy

- 1-WL $\rightarrow$ 2-WL $\rightarrow$ 3-WL ... form a hierarchy
- k-WL compares k-tuples instead of individual vertices
- Increasing k gives strictly more power
- But runtime grows rapidly: $O(n^k)$ per round

Beyond 1-WL: Higher WL Hierarchy

- 1-WL $\rightarrow$ 2-WL $\rightarrow$ 3-WL ... form a hierarchy
- k-WL compares k-tuples instead of individual vertices
- Increasing k gives strictly more power
- But runtime grows rapidly: $O(n^k)$ per round

Trade-off: expressiveness vs scalability

Treewidth and WL Power

- k-WL can distinguish graphs with treewidth k
- 1-WL matches tree homomorphisms (treewidth-1)
- 2-WL relates to path homomorphisms and cycles

Treewidth and WL Power

- k-WL can distinguish graphs with treewidth k
- 1-WL matches tree homomorphisms (treewidth-1)
- 2-WL relates to path homomorphisms and cycles

The hierarchy mirrors increasing structural complexity

Graphons and Limit Theory

- In dense graph limits: we use graphons — analytic objects
- Homomorphism densities $t(F, G)$ extend naturally to graphons
- One can define fractional isomorphism for graphons as well

Graphons and Limit Theory

- In dense graph limits: we use graphons — analytic objects
- Homomorphism densities $t(F, G)$ extend naturally to graphons
- One can define fractional isomorphism for graphons as well

Interesting link between WL, hom counts, and real analysis!

Compact Graphs: Open Questions

- A graph is compact if all fractional automorphisms are actual automorphisms
- Full characterization of compact graphs is unknown
- Compact graphs satisfy: fractional iso $\Rightarrow$ actual iso

Compact Graphs: Open Questions

- A graph is compact if all fractional automorphisms are actual automorphisms
- Full characterization of compact graphs is unknown
- Compact graphs satisfy: fractional iso $\Rightarrow$ actual iso

Study of compactness ties algebra and symmetry deeply

Descriptive Complexity Perspective

- 1-WL C^2 logic
- k-WL C^{k+1} logic (with $k+1$ variables)
- Higher WLs simulate higher-order logic expressiveness

Descriptive Complexity Perspective

- 1-WL C^2 logic
- k-WL C^{k+1} logic (with $k+1$ variables)
- Higher WLs simulate higher-order logic expressiveness

Implication: WL sits at the boundary of tractable logical inference

Expressive Power in Machine Learning

- GNNs built on message passing are at most 1-WL expressive
- Models that exceed 1-WL:
  - Higher-order GNNs (e.g., 2-WL GNNs)
  - Graph transformers
  - Subgraph-based models

Expressive Power in Machine Learning

- GNNs built on message passing are at most 1-WL expressive
- Models that exceed 1-WL:
  - Higher-order GNNs (e.g., 2-WL GNNs)
  - Graph transformers
  - Subgraph-based models

Design of future ML models can draw directly from WL theory

Suggested Reading

- Cai, Fürer, Immerman (1992): Descriptive Complexity of WL
- Grohe (2017): Descriptive Complexity of Graphs
- Babai (2016): Graph Isomorphism in Quasipolynomial Time
- Shervashidze et al. (2009): Weisfeiler–Lehman Kernel
- Morris et al. (2021): WL Hierarchy in GNNs

Open Problems

- Which graphs are compact?
- Can tree homomorphism profiles be efficiently inverted?
- How to best extend WL to graphon setting?
- What is the minimal logic beyond WL that distinguishes all graphs?

Open Problems

- Which graphs are compact?
- Can tree homomorphism profiles be efficiently inverted?
- How to best extend WL to graphon setting?
- What is the minimal logic beyond WL that distinguishes all graphs?

These questions bridge combinatorics, logic, and algebra

Conclusion and Wrap-Up

- WL refinement = a central concept in modern graph theory
- Three perspectives:
  - Combinatorics: color refinement
  - Algebra: fractional isomorphism
  - Enumeration: tree homomorphisms
- Rich area of ongoing research across theory and ML

Acknowledgments

- This talk is based on a paper presented at Euler Circle
- Author: **Mueed Awais**
- Supervised by: **Rachana Madhukara** (MIT)
- Special thanks to:
  - Euler Circle Community
  - Simon Rubinstein-Salzedo (founder)

Final Summary

- Explored three views of graph similarity:
  1. Weisfeiler–Leman Color Refinement
  2. Fractional Isomorphism
  3. Tree Homomorphism Counts
- All three are equivalent for testing 1-WL equivalence

What You Should Remember

- 1-WL is a fast, powerful but limited tool
- Tree homomorphisms and fractional isomorphisms give deeper insight
- Algebra, logic, and combinatorics work together

What You Should Remember

- 1-WL is a fast, powerful but limited tool
- Tree homomorphisms and fractional isomorphisms give deeper insight
- Algebra, logic, and combinatorics work together

Use WL as a stepping stone — not an endpoint

Three Takeaways

1. WL isomorphism test — but often good enough
2. Fractional isomorphism is algebraic color refinement
3. Homomorphism counts reflect structure and complexity

Suggested Practice

- Try running WL on simple graphs by hand
- Implement 1-WL in Python using NetworkX
- Compare spectra and tree hom counts between small graphs
- Use RDKit or PyTorch Geometric to explore GNN behavior

Stay Curious!

WL is just the beginning.

Explore logic, algebra, symmetry, and learning through graphs.

Questions?

Thank You!

Questions, comments, or discussion?

Feel free to reach out: `mueed.awais@eulercircle.org`

Thank you for your attention!