The Logical Expressive Power of Graph Neural Networks

Carsten Lutz
Universität Leipzig

Graph Learning

Graph Learning is important subarea of machine learning:

- many real world data represented as graphs

  social networks, molecule graphs, recommender systems, knowledge graphs

- standard ML models such as CNNs and RNNs require serialization / ordering

  obscures graph structure and results in classifiers that are not isomorphism-invariant

Many ML approaches to graph learning have been explored, e.g. kernel methods:

extract a lot of features (number of certain paths, subgraphs, WL colors, etc)

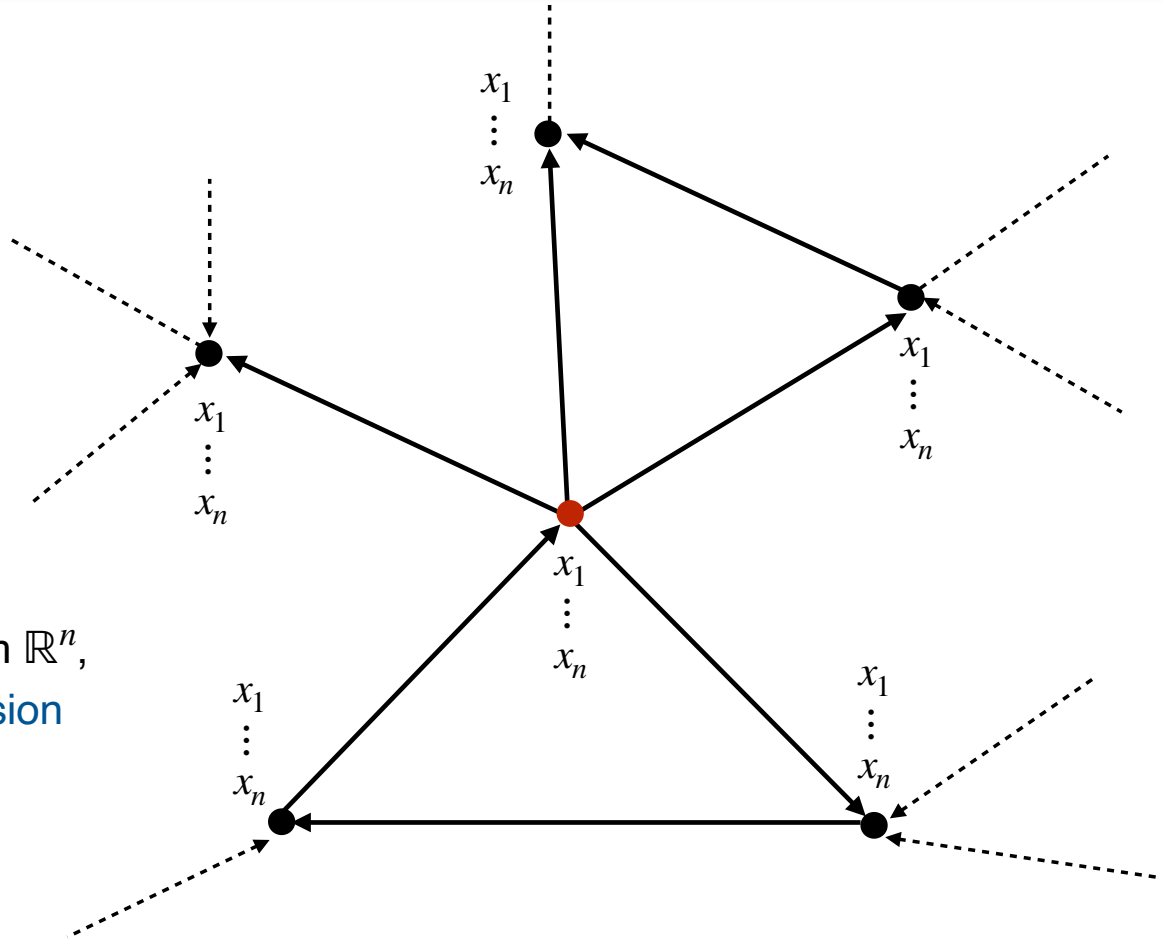
then use linear classifier such as SVM

In recent years, Graph neural networks (GNNs) [Scarselli et al. 2005] have taken the field by storm

Also known by other names: GCN, GraphSAGE, GIN, etc

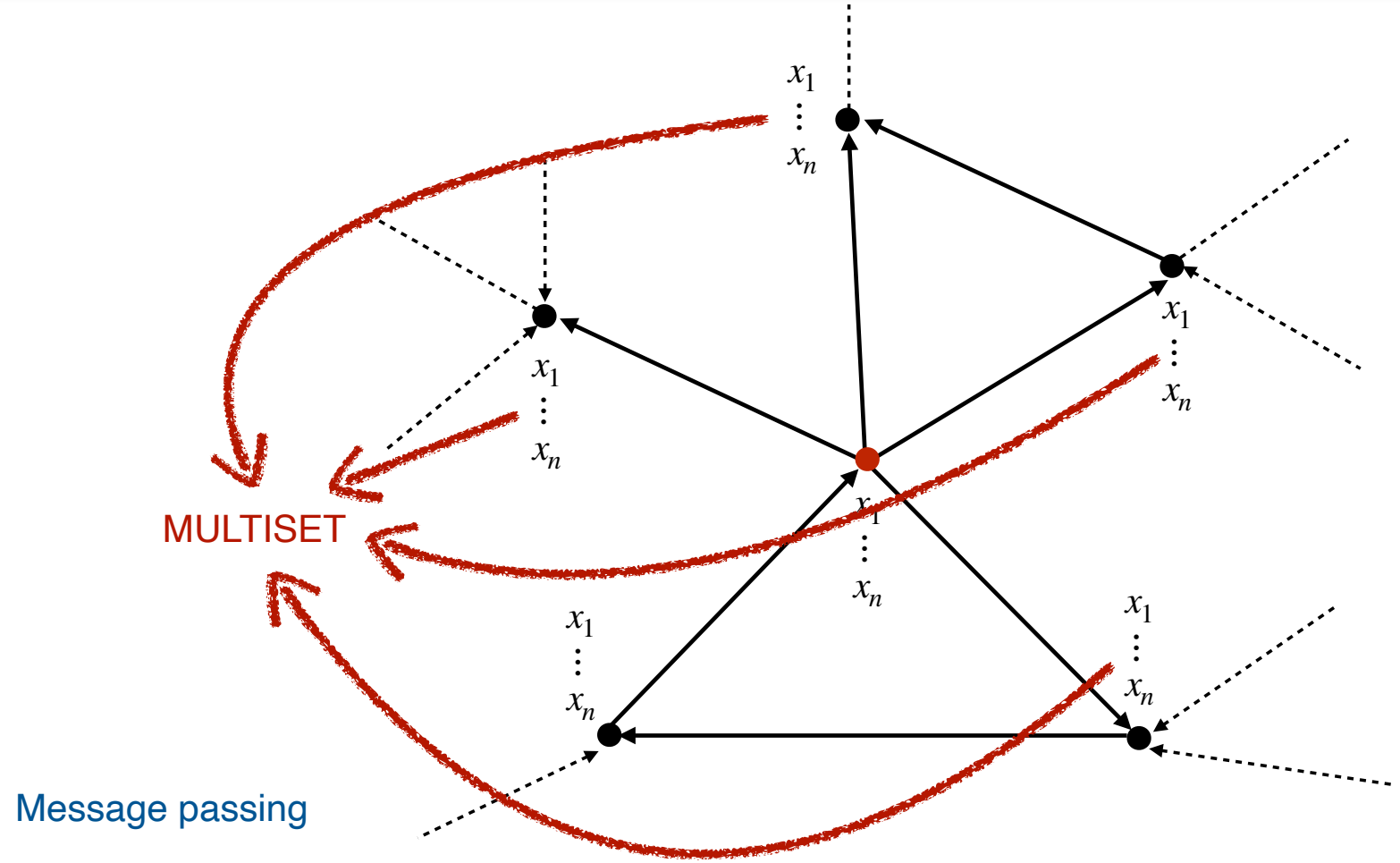
Interesting connections to KR!

Aggregate-Combine GNNs

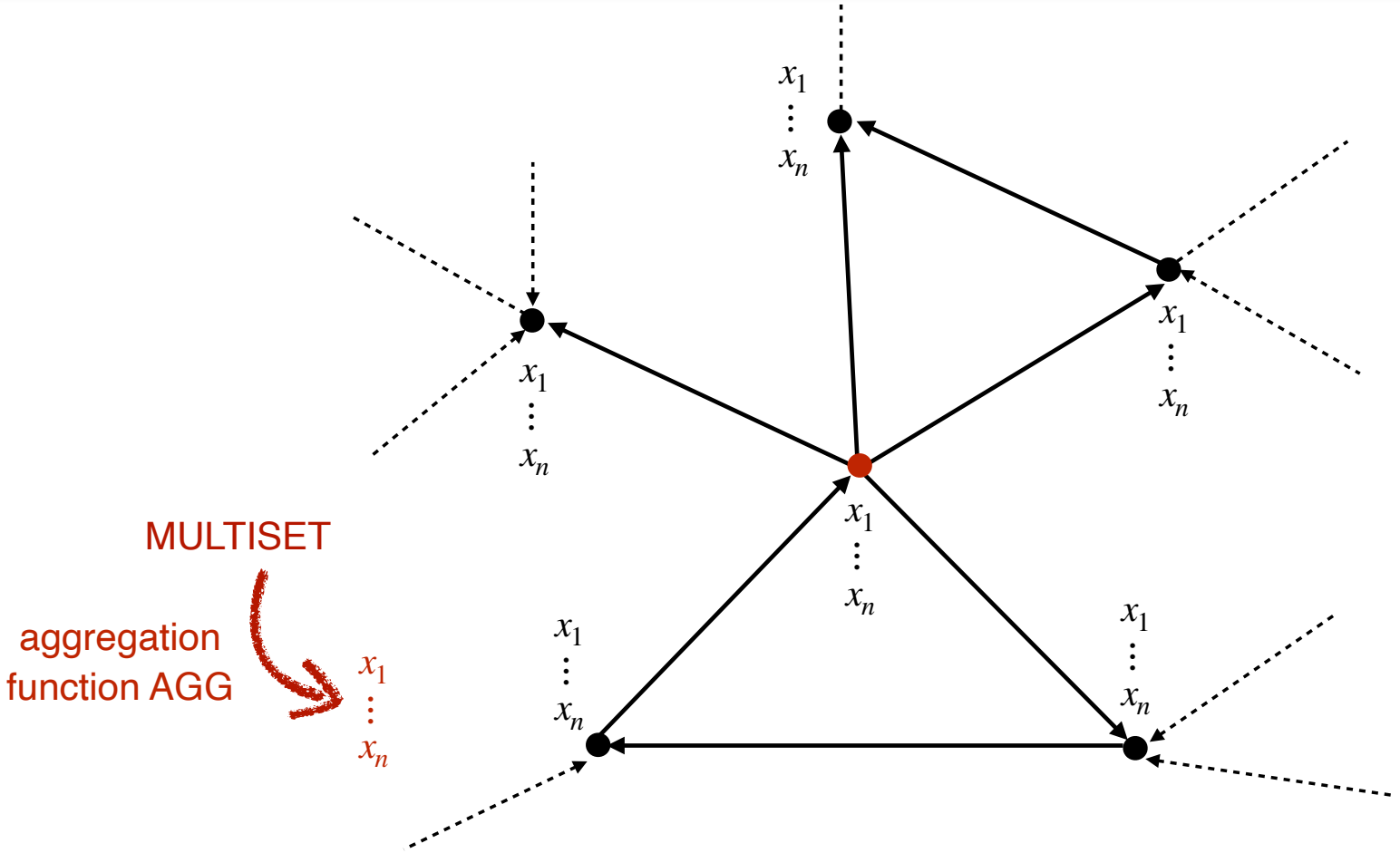


Embedding vectors from $\mathbb{R}^n$,
n internal GNN dimension

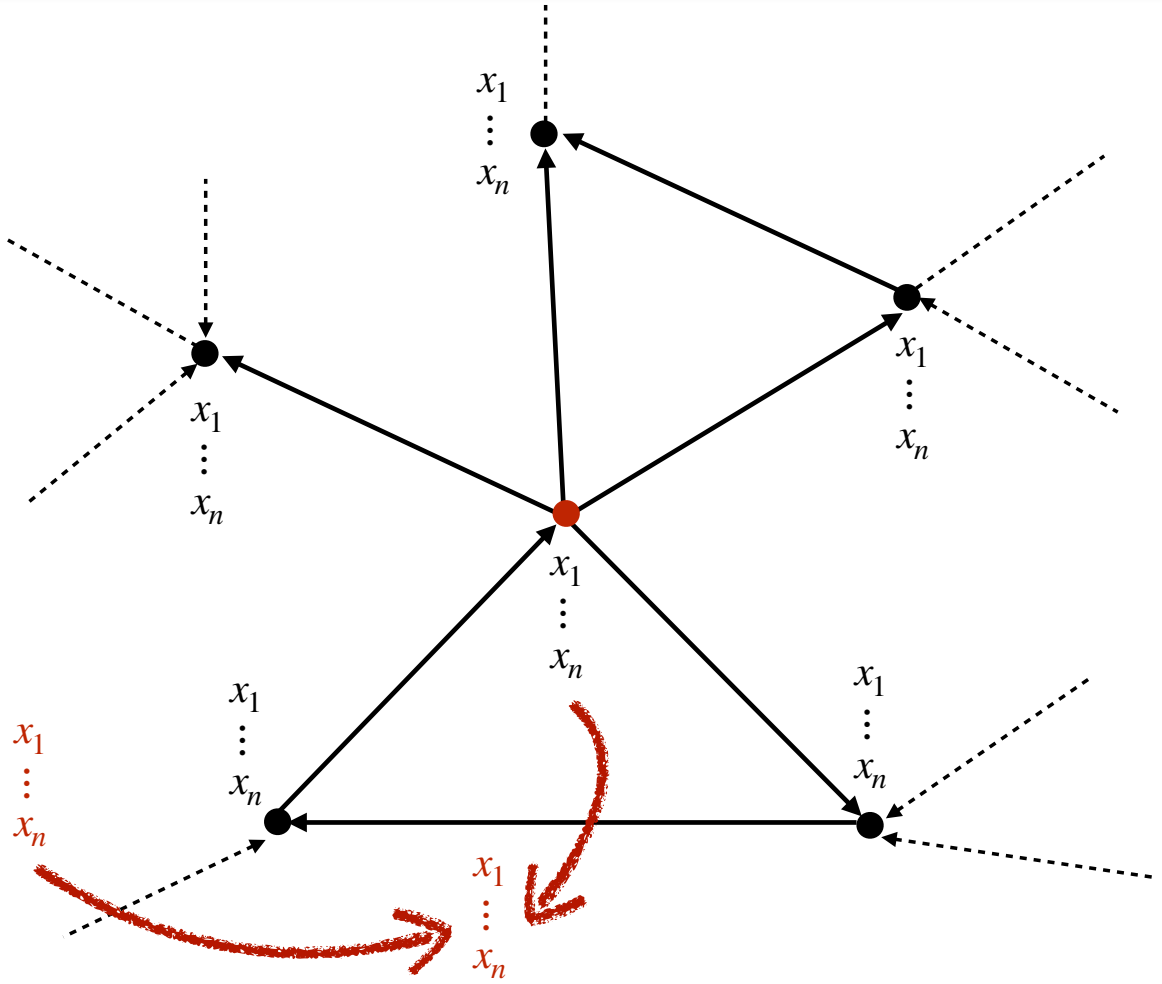
Aggregate-Combine GNNs



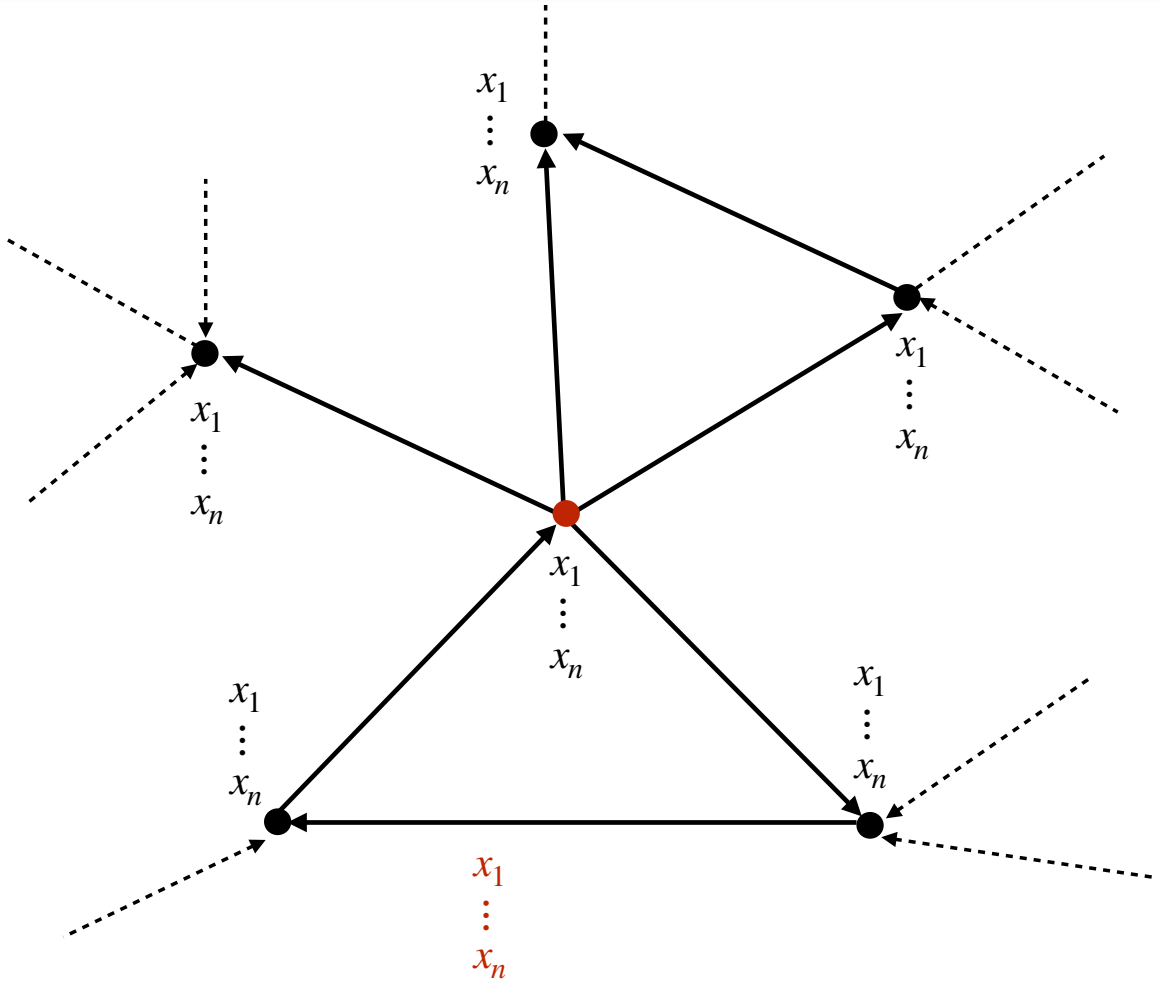
Aggregate-Combine GNNs



Aggregate-Combine GNNs



Aggregate-Combine GNNs



GNN transforms
vertex embeddings
into vertex embeddings

Aggregate-Combine GNNs

A basic case:

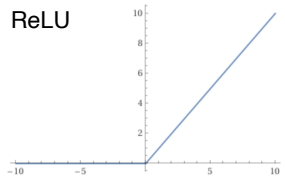
- constant number of iterations / layers
- classification function CLS maps final embedding to vertex classification (graph classification, link prediction, etc also possible)

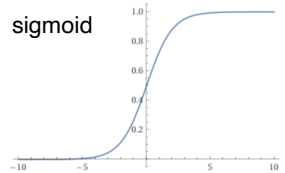
Many architectural choices:

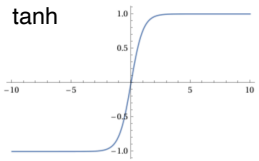
- aggregation function AGG: sum, mean, max (all applied component-wise)
- combination function COM represented by feedforward neural network (FFN)

  non-linear activation function ACT at every neuron, e.g. ReLU, sigmoid

  weights of neural network determined when GNN is learned (gradient descent)
- classification function CLS: often a simple threshold (but could also be FNN)





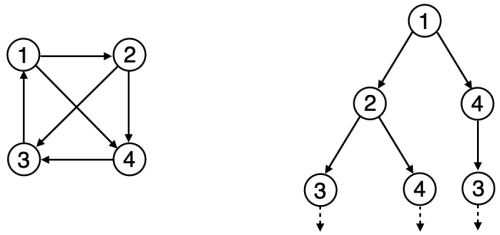


Expressive Power - First Hints

A natural question: Which vertex properties are expressible by a GNN, and which are not?

Two hints:

- message passing $\rightsquigarrow$ GNNs cannot distinguish between graph and unraveling



- multisets $\rightsquigarrow$ GNNs can count, i.e., distinguish different numbers of vertices with same vector

Oh, wait, that sounds like…

1-dimensional Weisfeiler-Leman test
($\approx$ colour refinement)

graded bisimulations

graded modal logic

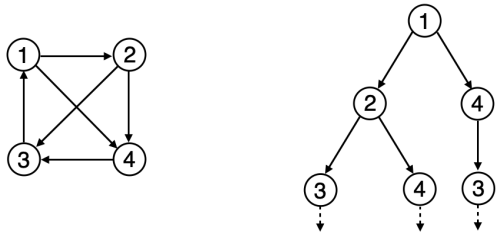
description logic
$\mathcal{ALCQ}$

Expressive Power - First Hints

A natural question: Which vertex properties are expressible by a GNN, and which are not?

Two hints:

- message passing $\rightsquigarrow$ GNNs cannot distinguish between graph and unraveling



- multisets $\rightsquigarrow$ GNNs can count, i.e., distinguish different numbers of vertices with same vector

Theorem [Morris et al. 2019/Xu et al. 2019]

If two vertices of a graph cannot be distinguished by the color refinement algorithm ($\approx$ 1-WL), then they cannot be distinguished by a GNN.

Equivalently: vertex properties definable by a GNN are invariant under graded bisimulation

Reminder: ML, DL, and Graded Bisimulation

Graded modal logic (GML): $\varphi, \psi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond^{\geq n}\varphi$ — "at least n successors satisfy φ"

Description logic $\mathcal{ALCQ}$: $C, D ::= A \mid \neg C \mid C \sqcap D \mid (\geq n\, r\, C)$

Modal logic (ML) / description logic $\mathcal{ALC}$: only $\Diamond^{\geq 1}\varphi$ / $(\geq 1\, r\, C)$ (no counting)

ML formulas are invariant under bisimulation, GML formulas under graded bisimulation

But even more than that:

Theorem [Van Benthem77 / Rosen97 / Otto2019]

For every FO-formula $\varphi(x)$, the following are equivalent (on finite graphs):

1. $\varphi(x)$ is invariant under graded bisimulation
2. $\varphi(x)$ is equivalent to a GML formula

The same is true for bisimulations and ML

GNN-GML connection first described by [Barcelo et al. 2019]

Where Everything is Nice: Max Aggregation

Theorem

Max-GNN = ML (= $\mathcal{ALC}$)

"$\subseteq$":

For n-layer Max-GNN, consider all finite trees of depth n

It is not hard to see that Max-GNNs are invariant under (non-graded!) bisimulations

ML can describe every finite tree up to bisimulation, by way of characteristic formulas

Only finitely many trees up to bisimulation $\rightsquigarrow$ take disjunction

"$\supseteq$":

Let φ be ML formula; ideas to build equivalent Max-GNN:

Embedding vectors: one position for every subformula ψ of φ

value 1 if ψ true at vertex and 0 if ψ false at vertex

One GNN layer per subformula, from simpler to more complex formulas

Where Everything is Nice: Max Aggregation

Combination function of the simple form $\mathrm{COM}(\bar{x}_1, \bar{x}_2) = \mathrm{ReLU}(\bar{x}_1 C + \bar{x}_2 A + \bar{b})$

(FFN without hidden layers / clipped affine function)

Negation: $V(\neg\psi) = (V(\psi) \cdot -1) + 1$

$$\bar{x}_1 = \begin{pmatrix} \\ \vdots \\ j \end{pmatrix}^{\psi} \cdot C = {}_j\begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \end{pmatrix}^{\ell} + \bar{b} = \begin{pmatrix} 1 \\ \vdots \\ \ell \end{pmatrix} = \begin{pmatrix} \\ \vdots \\ \ell \end{pmatrix}^{\neg\psi}$$

Modal Diamond:

$$\bar{x}_2 = \begin{pmatrix} \\ \vdots \\ j \end{pmatrix}^{\psi} \cdot A = {}_j\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}^{\ell} + \bar{b} = \begin{pmatrix} 0 \\ \vdots \\ \ell \end{pmatrix} = \begin{pmatrix} \\ \vdots \\ \ell \end{pmatrix}^{\Diamond\psi}$$

Componentwise max of feature vectors of neighbors

= 1 if some neighbor satisfies φ

Sum Aggregation

Sum aggregation: in $\Diamond\psi$ case now sum of 1's rather than max of 1's $\rightsquigarrow$ counting / GML

Theorem

With ReLU activation,

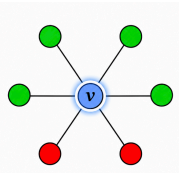
1. Sum-GNN $\supseteq$ GML ($= \mathcal{ALCQ}$)
2. Sum-GNN = GML on graphs of degree bounded by some constant d

Bounded degree removes the following problem:

For graded bisimulations, there are infinitely many distinguishable trees of any constant depth

With unbounded degree, Sum-GNNs are strictly more expressive than GML

e.g.: vertices with more green successors than red successors



GML-expressible on graphs of degree bounded by d: $\bigvee_{0 \le i < j \le d} \Diamond^{=i}\text{red} \wedge \Diamond^{=j}\text{green}$

Sum Aggregation

Sum aggregation: in $\Diamond\psi$ case now sum of 1's rather than max of 1's $\rightsquigarrow$ counting / GML

Theorem

With ReLU activation,

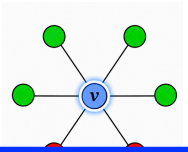
1. Sum-GNN $\supseteq$ GML ($= \mathcal{ALCQ}$)
2. Sum-GNN = GML on graphs of degree bounded by some constant d

Bounded degree removes the following problem:

For graded bisimulations, there are infinitely many distinguishable trees of any constant depth

With unbounded degree, Sum-GNNs are strictly more expressive than GML

Yet, we have the following (remember van Benthem!):



Theorem [BarceloEtAl2019]

Sum-GNN $\cap$ FO = GML (and in fact Any-GNN $\cap$ FO $\subseteq$ GML)

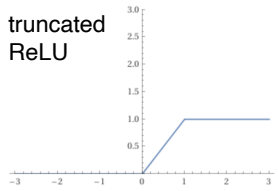
Sum Aggregation

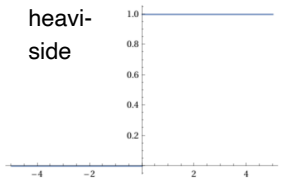
Sometimes, an absolute characterization is also attainable

Presburger modal logic: modal navigation + counting + linear arithmetic

E.g. at least twice as many green successors as red successors:

$$\varphi(x) \;:=\; \#\big[E(x,y) \wedge \mathsf{green}(y)\big] \;\geq\; 2 \cdot \#\big[E(x,y) \wedge \mathsf{red}(y)\big]$$





Theorem [Benedikt et al. 2024]

Sum-GNN = Presburger Modal Logic when coefficients are rational and activation functions are eventually constant (e.g. truncated ReLU, heaviside)

Some ideas about the construction Sum-GNN $\rightsquigarrow$ PML:

- proceed layer by layer (rather than `one-shot' as for Max-GNNs)
- show that only finitely many embedding vectors can occur on each layer, across all graphs
- for each vector, use PML formula to describe the (infinitely many) trees that give it

Sum Aggregation

Sometimes, an absolute characterization is also attainable

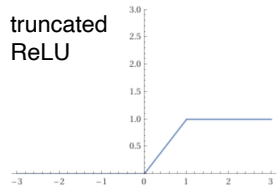
Presburger modal logic: modal navigation + counting + linear arithmetic

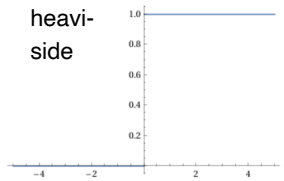
E.g. at least twice as many green successors as red successors:

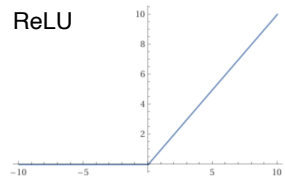
$$\varphi(x) \;:=\; \#\big[E(x,y) \wedge \mathsf{green}(y)\big] \;\geq\; 2 \cdot \#\big[E(x,y) \wedge \mathsf{red}(y)\big]$$

Theorem [Benedikt et al. 2024]

Sum-GNN = Presburger Modal Logic when coefficients are rational and activation functions are eventually constant (e.g. truncated ReLU, heaviside)







With non-truncated ReLU, the expressive power increases further, e.g.:

number of length-2 paths to green vertex
= number of length-2 paths to red vertex

An exact logical characterization remains an open problem—ML+C is upper bound [Grohe24]

Mean Aggregation

Arithmetic mean is popular aggregation function in practice because it admits sampling

Mean aggregation: in $\Diamond\psi$ case now mean of 1's and 0's over all successors

Consequence:

- "absolute counting" (e.g. at least 3 red successors) no longer possible
- "relative counting" (e.g. at least 1/2 of the successors red) still possible

Ratio modal logic (RML): $\varphi, \psi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond^{\geq r}\varphi$ where $r \in [0, 1]$

$G, v \models \Diamond^{\geq r}\varphi$ iff fraction of successors u of v with $G, u \models \varphi$ is at least r

Theorem [SchönherrL__ 2026]

With ReLU activation,

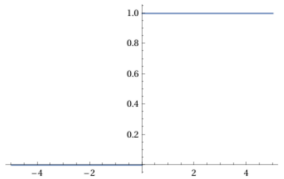
1. Mean-GNN $\supseteq$ RML
2. Mean-GNN = RML on graphs of degree bounded by some constant d

Mean Aggregation

Relative counting not expressible in FO, so the following might be expected:

Theorem [SchönherrL__ 2026]

Mean-GNN $\cap$ FO $=$ ML



Maybe less expected: ML $\rightsquigarrow$ Mean-GNN requires discontinuous activation (heaviside)

(despite some neighbors having value 1, mean can get arbitrarily close to 0)

For gradient descent, backpropagation, etc one wants continuous (even differentiable) functions

Theorem [SchönherrL__ 2025]

Mean-GNN $\cap$ FO $=$ AFML with continuous activation functions

AFML: alternation-free modal logic; no mixing of $\Diamond$ and $\Box$ e.g. $\Diamond p \wedge \Box q$ not expressible

No such effect for sum-GNNs and max-GNNs! Absolute case open, see also [Sälzer et al KR26]

So Many Dimensions…

random initialization
≈ order (with high probability)

edge labels / directions
≈ roles / inverse roles

aggregation function
≈ type of modality

activation function

graph transformer layers
≈ global modality

global readout

? ?
≈ counting global modality

hierarchical pooling

target-directed MP
≈ modal vs. guarded

classification function

positional / structural encodings
≈ extra predicates

continuity

reals vs. rationals vs. floats

recurrence
≈ fixpoints / modal substitution calculi

Is this useful beyond understanding expressive power?

Can KR contribute?

Three Facets

- Logical reasoning for static analysis and verification of GNNs
- Explanation of GNN Classifiers and Predictions
- Logic as a yardstick for developing GNNs of increased expressive power

Static Analysis and Verification

Static analysis problems: satisfiability, containment and equivalence (all finite!)

E.g.: GNN satisfiable if it classifies at least one vertex in one (finite) graph as true

Theorem [Benedikt et al. 2024]

For Sum-GNNs with rational coefficients and piecewise linear activation functions (e.g. ReLU), the static analysis problems are decidable and (co)NExp-complete.

But turns undecidable e.g. when global readout or incoming aggregation is added, also on undirected graphs

Theorem [Sälzer et al. 2025]

For Sum-GNNs with floating point coefficients / arithmetic, and PSpace-computable activation functions, the static analysis problems are decidable and PSpace-complete.

Adding global readout preserves decidability

(but increases complexity back to NExp) [Sälzer et al. KR 2026]

Static Analysis and Verification

Static analysis problems: satisfiability, containment and equivalence (all finite!)

E.g.: GNN satisfiable if it classifies at least one vertex in one (finite) graph as true

Theorem [Benedikt et al. 2024]

For Sum-GNNs with rational coefficients and piecewise linear activation functions (e.g. ReLU), the static analysis problems are decidable and (co)NExp-complete.

But turns undecidable e.g. when global readout or incoming aggregation is added, also on undirected graphs

Theorem [Sälzer et al. 2025]

For Sum-GNNs with floating point coefficients / arithmetic, and PSpace-computable activation functions, the static analysis problems are decidable and PSpace-complete.

More realistic verification task:

is every input vertex that satisfies φ transformed by the GNN to a vertex that satisfies ψ?

Explanation

Two types: (i) “What did this GNN learn” vs. (ii) “Why did this GNN classify this vertex as positive?”

Classical ML/GNN approach concerns (ii): identify small subgraph responsible for classification

...but of course using an $\mathcal{ALCQ}$ concept is a great alternative :)

Most work in logic concerns (i):

- For Sum-and Mean-GNNs, no unconditional logical characterization available

  May resort to sound explanations: concepts that `imply’ the GNN classifier

  Case study [MorrisEtAl2024]:

  Focus on $\mathcal{ELU}$ explanations: simple monotone language ($\wedge$,$\vee$, $\exists$)

  Result: sound $\mathcal{ELU}$ explanations rarely exist even if supported by dataset (in 0 cases!)

Explanation

Two types: (i) “What did this GNN learn” vs. (ii) “Why did this GNN classify this vertex as positive?”

Classical ML/GNN approach concerns (ii): identify small subgraph responsible for classification

...but of course using an $\mathcal{ALCQ}$ concept is a great alternative :)

Most work in logic concerns (i):

- For Sum-and Mean-GNNs, no unconditional logical characterization available

  May resort to sound explanations: concepts that `imply’ the GNN classifier

- Max-GNN $= \mathcal{ALC}$, but extraction of equivalent concept expensive [TenaCucalaEtAl24]

  Type (ii) explanations, in contrast, may still be practical

Explanation

Two types: (i) “What did this GNN learn” vs. (ii) “Why did this GNN classify this vertex as positive?”

Classical ML/GNN approach concerns (ii): identify small subgraph responsible for classification

...but of course using an $\mathcal{ALCQ}$ concept is a great alternative :)

Most work in logic concerns (i):

- For Sum-and Mean-GNNs, no unconditional logical characterization available

  May resort to sound explanations: concepts that `imply’ the GNN classifier

- Max-GNN $= \mathcal{ALC}$, but extraction of equivalent concept expensive [TenaCucalaEtAl24]

- One may force the GNN to be monotonic, thus enable $\mathcal{ELU}$-explanations: [TenaCucalaEtAl22]

  Non-negative weights, max aggregation, monotone and non-negative activation

  Appears to work well in practice (but restricts GNN expressiveness)

More Expressive GNN Models

Message passing fails to identify even simple graph structures: triangles, cliques, etc

Practitioner's response: use positional / structural encodings

Enrich vertices with e.g. subgraph counts, Laplacians, WL colors, random walk features, etc

More principled ways forward have been proposed, e.g. using WL as yardstick

From 1-WL to k-WL: Higher-order GNNs [Morris et al 2019]:

- Embedding vectors associated with sets of vertices of fixed size k
- Initial embedding vector derived from isomorphism type of the set
- Message passing between sets that differ in exactly one vertex

But of course logic can also act as a yardstick (and in fact k-WL = C^{k+1})

More Expressive GNN Models

Many different approaches:

- Lift the objects:
  higher-order GNNs, tensor GNNs, cellular/simplicial GNNs
- Change the neighborhoods:
  subgraph GNNs, nested GNNs, ego-GNNs, path GNNs
- Add global structure:
  graph transformers, virtual nodes, relational pooling
- Use query / homomorphism patterns:
  deep homomorphism networks, template GNNs, hypergraph/relational GNNs

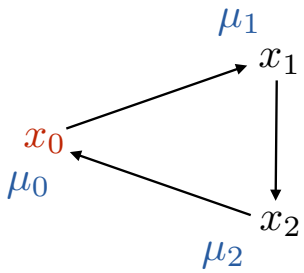
Deep Homomorphism Networks

Higher-Order GNNs are computationally expensive, and there are alternatives:

- associate embedding vectors with vertices
- build in graph patterns that you deem important for your application

$\rightsquigarrow$ Deep Homomorphism Networks (DHNs)

For instance:



homomorphism gives embedding vector e_i for each variable x_i

passed through (learned) transformation function μ_i

product $\mu_0(e_0) \cdot \mu_1(e_1) \cdot \mu_2(e_2) \rightsquigarrow$ multiset

Brings together GNNs and conjunctive queries, attractive for database applications (+ hom counts)

Theorem [SchönherrTenCateFunkKimelfeldL___Soeteman 2026]

With ReLU activation,

1. Sum-DHN $\supseteq$ UNFOC (= unary negation fragment of FO + counting quantifiers)
2. Sum-DHN = UNFOC on graphs of degree bounded by some constant d