Lecture Notes on Verifying Graph Neural Networks

François Schwarzentruber
ENS de Lyon

August 7, 2026

Preface

In these lecture notes, we first recall the connection between graph neural networks, Weisfeiler-Lehman tests and logics such as first-order logic and graded modal logic. We then present a modal logic in which counting modalities appear in linear inequalities in order to solve verification tasks on graph neural networks. We describe an algorithm for the satisfiability problem of that logic. It is inspired from the tableau method of vanilla modal logic, extended with reasoning in quantifier-free fragment Boolean algebra with Presburger arithmetic.

Fall 2025

In order to give an idea to lecturers that may use this material, this course was taught at a master level at ENS de Lyon. The lecture was divided in four 2-hour sessions + mid-exam as follows:

1. 29th of September 2025: definition of GNN, color refinement, FO, FO2, FOC, FOC2, FO2 is NEXPTIME-hard, correspondence between FOC2 and color refinement

2. 3th of October: discussion on the correspondence, ML, standard translation, GML, correspondence between GML and color refinement, tableau method for ML, satisfiability problem ML is in PSPACE. (soundness and completeness of the tableau method was not proven)

3. 6th of October: satisfiability problem ML is PSPACE-complete. GNN with truncReLU and $\mathrm{K}^{\#}$. GNN with truncReLU and $\exists PA^*$ (only the statement).

4. 10th of October: GNN with truncReLU and $\exists PA^*$. Issue with the satisfiability procedure. QFBAPA and $\mathrm{K}^{\#}$ full satisfiability procedure.

5. 13th of October: mid-exam, followed by a discussion session

ESSLLI 2026

We plan the following program for ESSLLI 2026 from 3th to 7th of August 2026. The lecture is divided in five 90-minute sessions as follows:

1. Monday: motivation and definition of GNN; color refinement

2. Tuesday: Panaroma of logics to talk about graphs: FO, FOC (FO with counting), ML (modal logic), GML (graded modal logic). Correspondence between GML and color refinement. Correspondence between GML and GNNs. EF games.

3. Wednesday: Pebble games. correspondence between FOC_2 and color refinement. GNNs and $\mathrm{K}^{\#}$, a logic with Presburger arithmetics.

4. Thursday: Satisfiability of $\mathrm{K}^{\#}$. Recall of tableau method for modal logic. Carathéodory bound. Tableau method for $\mathrm{K}^{\#}$.

5. Friday: Medley of different topics

Fall 2026

This material will be reused for a course at ENS de Lyon again!

Acknowledgments

First I want to thank Pierre Nunn: we started to work on GNNs and logic in 2022. I want to warmly thank Artem Chernobrovkin, Avijeet Ghosh, Marco Sälzer and Nicolas Troquard for all the discussions that helped to improve these lecture notes. I also warmly thank the master students (M2 level) at ENS de Lyon for their comments during the lecture.

Contents

Chapter 1

Graph neural networks

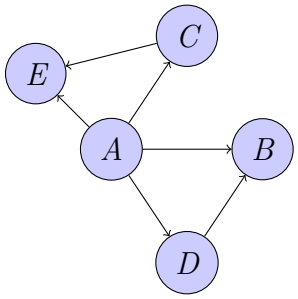
Key references: [Sato, 2020], [Grohe, 2021]

1.1 Motivation

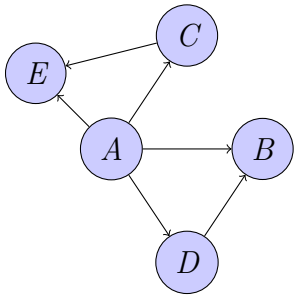
1.1.1 Why graphs?

Graphs are everywhere. A directed (resp. undirected) *graph* is a set of *vertices* with *edges*. To be more general we consider directed graphs (undirected graphs are just directed graphs where each undirected edge is replaced by a pair of directed edges). Some of the examples are undirected graphs.

Example 1. *Here is an example of directed graph (e.g. a social network with a 'I follow you' relation):*

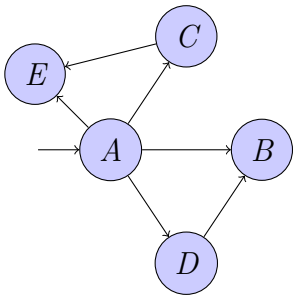


Example 2. *Here is an example of undirected graph (e.g. modelling a molecule):*



Example 3. *Here is an example of pointed directed graph (e.g. a person in a social network with a 'I follow you' relation):*

Figure 1.1: A three-voice clausula vera from Palestrina's Magnificat Secundi Toni: Deposuit potentes, mm. 27–28. And its graph representation with three types of edges: notes beginning at the same moment, notes that are following each other, notes that are overlapping.



1.1.2 Why machine learning?

In many applications, it is difficult to formalize and write properties on graphs that correspond to some need. That is why, graph neural networks have been proposed as machine learning models in [Scarselli et al., 2009] for classifying graphs. They are used in various applications:

- recommendation in social network [Salamat et al., 2021],
- knowledge graphs [Ye et al., 2022],
- chemistry [Reiser et al., 2022],
- drug discovery [Xiong et al., 2021]
- voice separation in music scores [Karystinaios et al., 2023], see Figure 1.1
- Heuristics in epistemic planning [Briglia et al., 2025] etc.

They are used for *classification* (yes/no questions) or *regression* (guess a value) for a given graph or a pointed graph (a vertex in some graph)[1].

The following table gives some examples of informal questions:

	classification	regression
on graphs	does the graph represent a toxic molecule?	what is the temperature of fusion of a given molecule represented by a graph?
on pointed graphs	do we recommend some person?	what is the price of some furniture?

[1] In the literature, they may say 'on graph-nodes' or 'on nodes'.

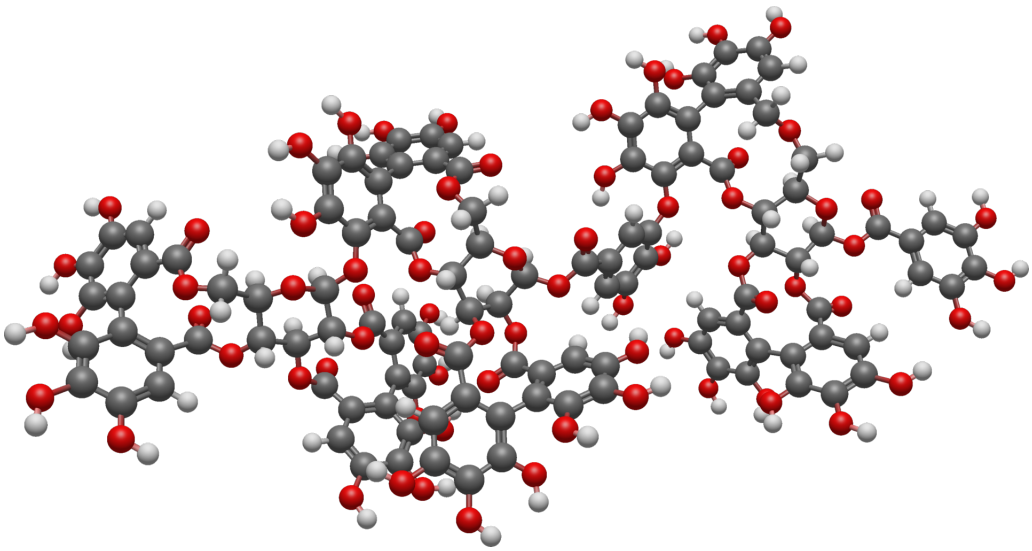
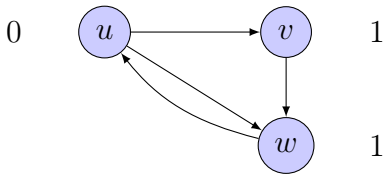


Figure 1.2: The molecule of Raspberry ellagitannin.

1.1.3 Why not just using standard neural networks?

A standard neural network is function from $\mathbb{R}^d$ into $\mathbb{R}$. It takes a vector of numbers as an input and produces a number. The simple answer is: we can use standard neural networks. The input graph, for instance,



is represented by the adjacency matrix

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

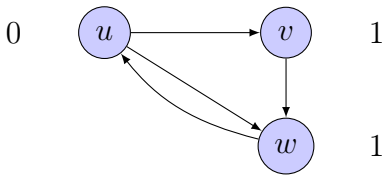
and the vector

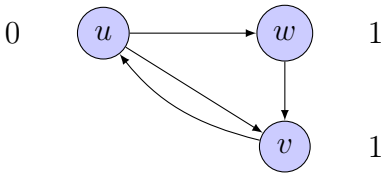
$$\begin{pmatrix} 0 & 1 & 1 \end{pmatrix}$$

for the feature values at each vertex. Then a neural network could just take a 'super' vector that contains all the values in the adjacency matrix and the vector of feature values. However, that methodology suffer from two drawbacks.

First graphs can be *arbitrary large*. For instance, large molecules, large music scores, large social networks. Would you really learn to classify up to size 3, 4, 1000, and not being able to generalize to large graphs (see Figure 1.2)?

Second, two isomorphic graphs are just the same. The neural network should give the same output for the two following isomorphic graphs





despite their adjacency matrix are not the same (because we consider another order of the vertices):

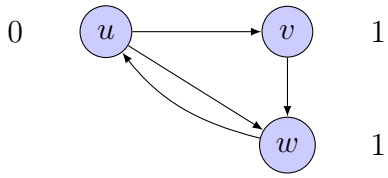
$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

1.2 Graph Neural Networks

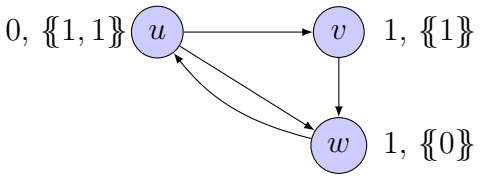
A graph neural network is a machine learning that is *independent* from the size of the graph, and that gives the same answer to *isomorphic* graphs. Let us explain.

1.2.1 Informal Definition

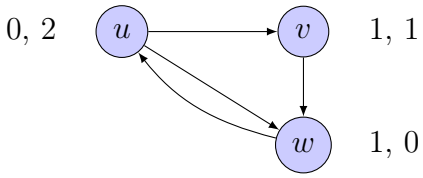
Inputs to classify are *labelled* graphs. For instance:



A graph neural networks works via *message-passing* mechanism as follows. First each vertex receives feature values from their neighbours/successors.



We wrote values from the neighbours/successors as *multisets* because values can be repeated if several successors have the same values. Then we apply a so-called *aggregation* function, typically the *sum* on the received values. We obtained the following aggregated values:

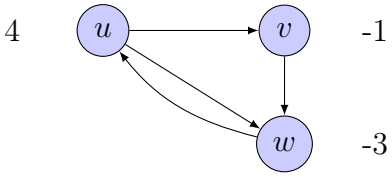


After that, we apply a *combination* function, typically a standard *neural network*. For simplicity we suppose that this phase is first a linear function then the application of an

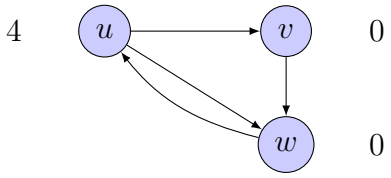
activation function. In our example, we suppose that the linear function applied in each vertex is:

$$-3 \times \text{current value} + 2 \times \text{aggregated value}$$

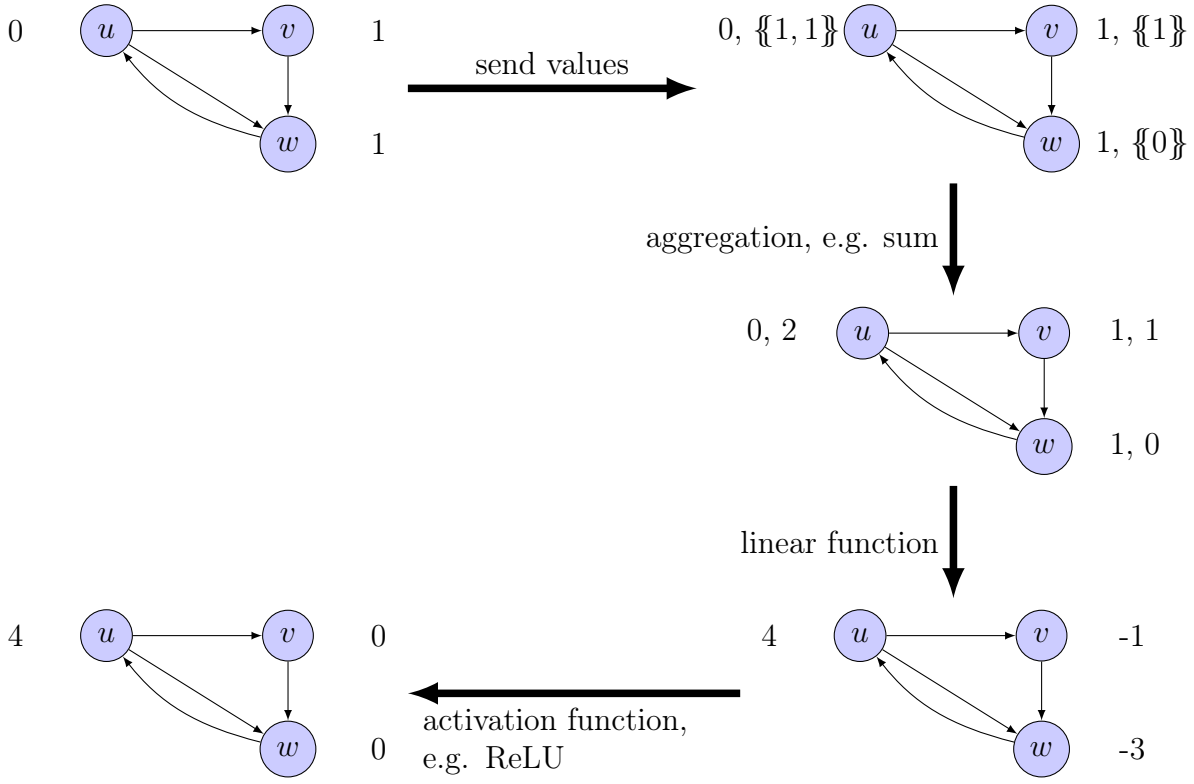
We get:



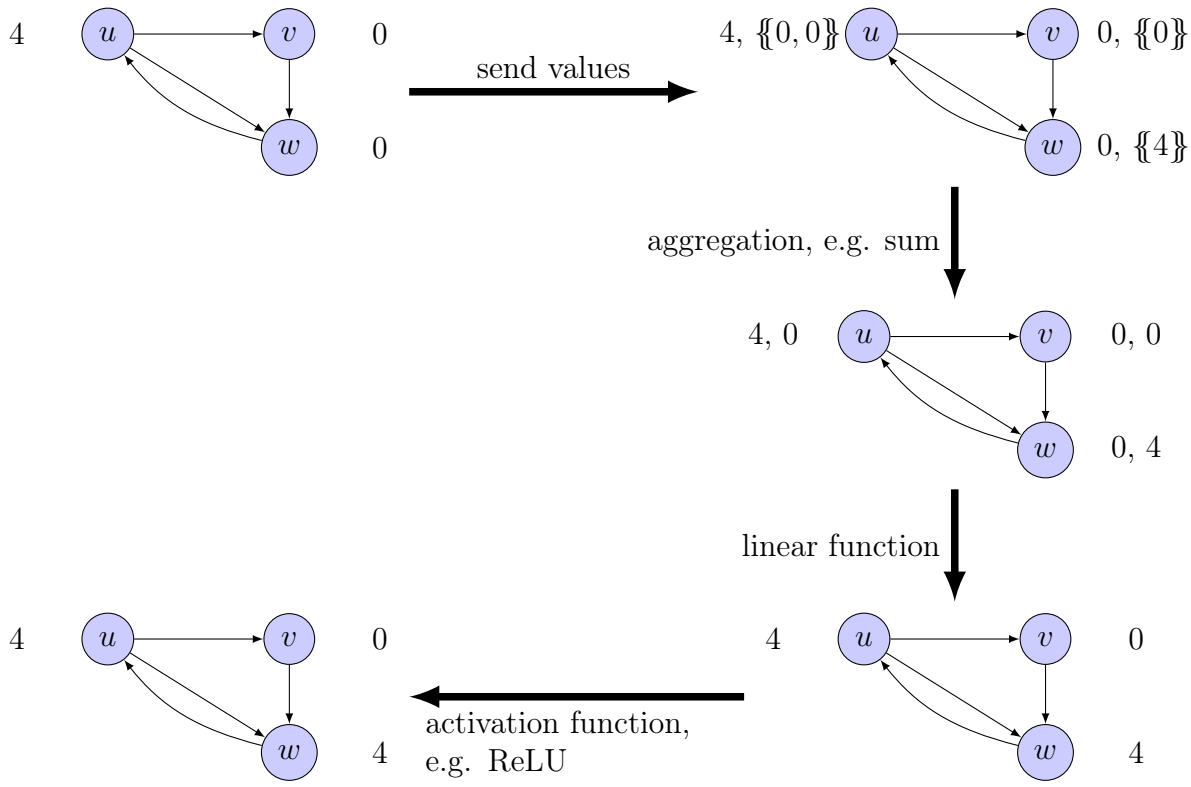
Secondly we apply an activation function, for instance ReLU which replaces each negative value by 0 while keeping positive values. We obtain:



The process we saw so far is called a *layer* and can be summed up as follows.



A GNN consists in applying several layers. The resulting labelled graph of the first layer is the input of the second layer, and so on. A second layer where is the linear function is let say current value + aggregated value is then:



For classifying there are different options. Either we do (binary) *node classification*. In that case, a linear inequality is checked on the vertex. For instance,

$$\text{feature value} \geq 1.$$

For instance, as the feature value at u (i.e. 4) is greater or equal than 1, the pointed graph G, u is classified positively. Here, G, v is classifed negatively since $0 \geq 1$ does not hold.

Or we do (binary) *graph classification*. In that case, a possibility is to perform a *global readout* aka *global aggregation*. Here, $4{+}0{+}4 = 8$. A linear inequality enables to classify the graph as a whole. For instance

$$\text{final value} \geq 1.$$

Here as $8 \geq 1$, the graph is classified positively.

The very choice of aggregation function (sum/max/mean, etc.), the activation function, combination function, global readout at the end or at each step depends on the *architecture* of the GNN we use.

1.2.2 Formal Definition

We consider labelled directed graphs $G = (V, E, \ell)$ where V is a non-empty finite set of vertices, $E \subseteq V \times V$ is a set of edges and $\ell : V \to \mathbb{R}^d$ is a labelling function, i.e. each vertex u is labelled with a vector $\ell(u)$ containing d real numbers. We denote by $E(u)$ the set of successors of u. Formally, $E(u) = \{v \in V \mid (u, v) \in E\}$. We encode a standard *Kripke structure* by a labelled graph $G = (V, E, \ell_0)$ with $\ell_0 : V \to \{0, 1\}^d$.

A GNN N is usually defined as a tuple of parameters (see [Barceló et al., 2020], [Nunn et al., 2024], [Benedikt et al., 2024]). To make it more concrete, we present it as an algorithm (parametrized by the weights computed during some learning process), see Figure 1.3. It takes as an input

a pointed graph made up of a labelled graph $G = (V, E, \ell_0)$ and a vertex u. It outputs yes/no.

main function $N((V, E, \ell_0), u)$
  $\ell_1 := layer_1(V, E, \ell_0)$
  $\ell_2 := layer_2(V, E, \ell_1)$
  $\vdots$
  $\ell_L := layer_L(V, E, \ell_{L-1})$
  return yes **if** $w^t \ell_L(u) + b \geq 0$ **else** no

function $layer_i(V, E, \ell)$
  $\ell' :=$ new labelling $V \to \mathbb{R}^d$
  for vertices $u \in V$ **do**
    $\ell'[u] := \vec{\alpha}(A_i \times \ell[u] + B_i \times \sum \{\!\{\ell[v] \mid v \in E(u)\}\!\} + b_i)$
  return ℓ'

Figure 1.3: A graph neural network N presented as an algorithm. The main function is N. It computes a sequence of labellings $\ell_1, \ell_2, \ldots, \ell_L$ via functions $layer_i$. Learnt weights are w, b, A_i, B_i, b_i.

Definition 4 (graph neural network)**.** *A GNN is an algorithm as shown in Figure 1.3.*

A GNN N computes a sequence of labellings $\ell_1, \ell_2, \ldots, \ell_L$ via the application of so-called layers. For avoiding cumbersome notations, we suppose that all these labellings assign a vector of dimension d to each vertex (while in generality the dimension d may be different for each layer). In each layer $layer_i$, the function $\vec{\alpha} : \mathbb{R}^d \to \mathbb{R}^d$ is the point-wise application of an activation function $\alpha : \mathbb{R} \to \mathbb{R}$. The *activation function* α could be for instance $ReLU : x \mapsto \max(0, x)$ or $truncReLU : x \mapsto \max(0, \min(x, 1))$. Then $A_i \in \mathbb{R}^{d \times d}$ and $B_i \in \mathbb{R}^{d \times d}$ are matrices of weights, $\times$ is the standard matrix-vector multiplication, $b_i \in \mathbb{R}^d$ is a bias vector, $\{\!\{.\}\!\}$ is the multiset notation, and $\sum$ is the summation operation of all vectors in the multiset.

The ending linear inequality $w^t \ell_L(u) + b \geq 0$ where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are weights is used to classify the vertex u. In the sequel, the set of pointed labelled graphs positively classified by a GNN N is denoted by $[\![N]\!] := \{(G, u) \mid N(G, u) \text{ returns yes}\}$.

We can also present a GNN N that computes a sequence $N^{(0)}, N^{(1)}, \ldots$ of labellings:

- $N^{(0)}(G, u) = \ell_0(u)$;
- $N^{(t)}(G, u) = \vec{\alpha}(A_t \times N^{(t-1)}(G, u) + B_t \times \sum \{\!\{N^{(t-1)}(G, v) \mid v \in E(u)\}\!\} + b_t)$ for all $u \in V$.

1.3 GNN on graphs

For *graph classification*, the last operation could be:

return yes **if** $w \sum_{u \in V} \ell_L(u) + b \geq 0$ **else** no

The sum is taken over *all* vertices u in the graph. This kind of operation is *global readout*. We could also have this operation at each layer:

$$\begin{aligned} N^{(t)}(G,u) =& \vec{\alpha}(A_t \times N^{(t-1)}(G,u) \\ &+ B_t \times \sum \{\!\{ N^{(t-1)}(G,v) \mid v \in E(u) \}\!\} \\ &+ C_t \times \sum \{\!\{ N^{(t-1)}(G,v) \mid v \in V \}\!\} \\ &+ b_t). \end{aligned}$$

In order to keep the course simple, we postpone the discussion on global readout in Chapter 9.

1.4 Variants

There are variants for the update instruction:

$$N^{(t)}(G,u) := \vec{\alpha}(A_t \times N^{(t-1)}(G,u) + B_t \times \sum \{\!\{ N^{(t-1)}(G,v) \mid v \in E(u) \}\!\} + b_t).$$

1.4.1 Aggregation function

Up to now, we took the sum $\sum$ for aggregating the previous results on the neighbours. We say that used $\sum$ as an *aggregation function*. We could also max as an aggregation function:

$$N^{(t)}(G,u) := \vec{\alpha}(A_t \times N^{(t-1)}(G,u) + B_t \times \max \{\!\{ N^{(t-1)}(G,v) \mid v \in E(u) \}\!\} + b_t).$$

Examples of aggregation functions are: the sum $\sum$, the max, the min, the mean, weighted sums (where weights are given in the graphs, i.e. the edges in the input graph are weighted)

1.4.2 Combine function

The combine function *combine* explains how to fuse the previous value at the current node and the result of the aggregation on the neighbours. To sum up, we can write:

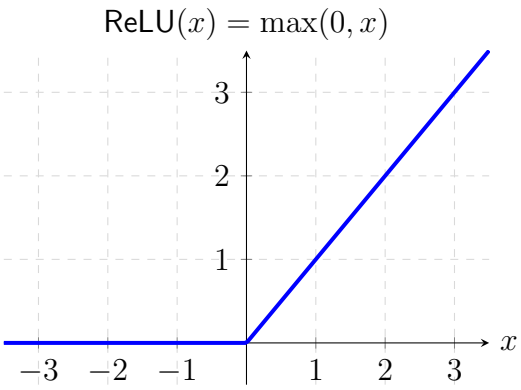
$$N^{(t)}(G,u) := combine_t(N^{(t-1)}(G,u), aggregation_t(\{\!\{ N^{(t-1)}(G,v) \mid v \in E(u) \}\!\})).$$

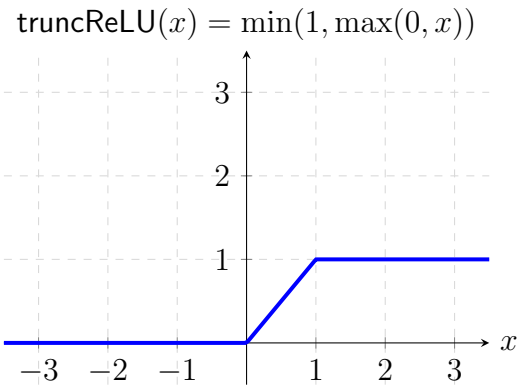
The combine function $combine_t$ can be a standard neural network, or can be a standard 1-layer neural network:

$$\alpha(A_t \times \text{current vector} + B_t \times \text{aggregated vector} + b_t).$$

1.4.3 Activation function

Activation functions are used to break the linearity (otherwise, we would live in a world of linear functions only...). Standard activation functions are ReLU, truncReLU, sigmoid, etc.





From now we consider AC-GNNs (aggregation-combine GNNs) where the aggregation function is sum (if not said otherwise), the combine function is a standard 1-layer neural network. In the same way, ACR-GNNs (aggregation-combine-global readout GNNs) are similar but also have global readout (global aggregation) at each layer:

$$N^{(t)}(G,u) := combine_t(N^{(t-1)}(G,u), aggregation_t(\{\!\{N^{(t-1)}(G,v) \mid v \in E(u)\}\!\}))$$
$$aggregation'_t(\{\!\{N^{(t-1)}(G,v) \mid v \in V\}\!\})).$$

Exercises

Exercise 1. *Give a ACR-GNN that write $|V|^2$ in some feature of some vertex, where $|V|$ is the number of vertices in the input graph.*

Exercise 2 (From [Wittig et al., 2026] p. 6)**.** Bellman-Ford algorithm *computes shortest path distances from a source s in a weighted directed graph G. We recall the update function of Bellman-Ford:*

$$d^{(t)}(v) := min(d^{(t-1)}(u), d^{(t-1)}(u) + weight_{uv} \mid uv \in E)$$

- *Explain how to implement a GNN that computes distances after t rounds; explain also how to encode the initial graph.*

Hint: add self loops uu with $weight_{uu} = 0$; initialize the $d^{(0)}(s) = 0$ and $d^{(0)}(u)$ to some bug number for all $u \neq s$.

Exercise 3 (From [Wittig et al., 2026] p. 5)**.** Page rank *is the algorithm used by Google to rank the webpages. Fix $\lambda \in [0, 1]$ called the damping factor. The algorithm works as follows. Initially,for all webpages u,*

$$PR^{(0)}(u) = \frac{1}{|V|}.$$

Then

$$PR^{(t)}(u) = (1 - \lambda) + \frac{\lambda}{outdeg_u} \sum_{v|uv \in E} PR^{(t-1)}(v)$$

where $outdeg_u$ is the out-degree of u i.e. $outdeg_u := \#\{v \mid uv \in E\}$.

- *Explain how to implement a GNN that computes page ranks after t rounds.*

Chapter 2

Weisfeiler-Lehman tests

Key references: [Sato, 2020], [Grohe, 2021]

2.1 Motivation

What can we compute/classify with graph neural networks? A layer computation is:

$$N^{(t)}(G,u) := combine_t(N^{(t-1)}(G,u), aggregation_t(\{\!\{N^{(t-1)}(G,v) \mid v \in E(u)\}\!\})).$$

To understand expressivity of GNNs, let us focus on the abstract syntactic computation by removing the very precise computation:

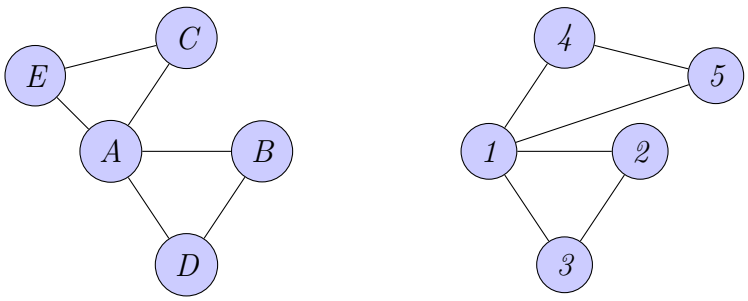
$$N^{(t)}(G,u) := \cancel{combine_t}(N^{(t-1)}(G,u), \cancel{aggregation_t}(\{\!\{N^{(t-1)}(G,v) \mid v \in E(u)\}\!\})).$$

The multiset corresponds to counting, while looking to successors only meaning that a AC-GNN gives the same answer to a graph and its *unravelling*. This is a reminiscence of Weisfeiler-Lehman test that is used a quick test for graph isomorphism.

2.2 Graph isomorphism

Two graphs are isomorphic if they are the same, up to vertex renaming. Here is a formal definition.

Example 5. *Here are two graphs that isomorphic:*



Definition 6. *Two labelled graphs $G = (V, E, \ell)$ and $G' = (V', E', \ell')$ are* isomorphic *if there exists a bijection $\pi : V \to V'$ such that:*

1. *for all vertices $u \in V$, $\ell[u] = \ell[\pi(u)]$*
2. *for all vertices $u, v \in V$, uEv iff $\pi(u)E'\pi(v)$.*

Graph isomorphism is in NP, but likely to be NP-complete, because then the polynomial hierarchy would collapse [Schöning, 1988]. Testing graph isomorphisms can be done in quasipolynomial time [Babai, 2016]: more precisely in time $exp((\log n)^{O(1)})$ where n is the number of vertices. Weisfeiler-Lehman tests are procedures running in poly-time[1] which do "almost" solve graph isomorphism.

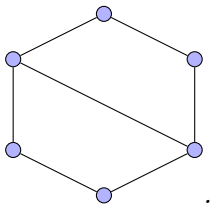
2.3 1-WL aka Colour refinement

2.3.1 Description

There are different presentations of 1-WL (1-Weisfeiler-Lehman) aka color refinement aka naive vertex refinement in the literature [Morgan, 1965]. Informally, it works as follows on a graph $G = (V, E, \ell_0)$.

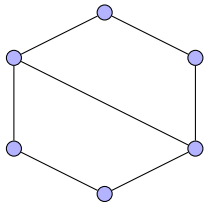
- Initially, colour each vertex v with the colour $\ell_0(v)$.
- Iteratively, recolour each vertex based on its current colour and the multiset of colours of its neighbours.
- Repeat until the colouring stabilizes.

2.3.2 Examples



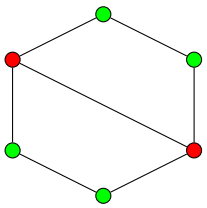
Example 7. *(from [Grohe, 2021]) Consider the graph* .

As it is unlabelled, all vertices have the same colour (let say blue):

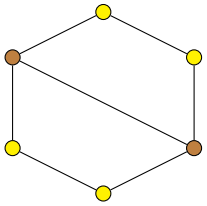


At the first step, only the degree of the vertices matter. We then give the same colour for vertices with the same degree:

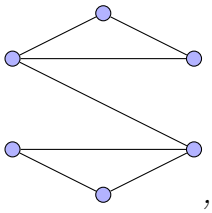
[1] It is possible to implement it in $O((|V| + |E|) \log |V|)$ [Cardon and Crochemore, 1982].

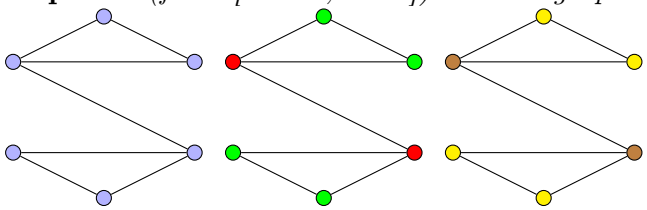


Now there are two types of vertices: the green ones with one red and one green neighbour (repainted in yellow), and the red ones with two green neighbours and one red (repainted in brown):



The colour differentiates in the same way the vertices. We say that the procedure stabilizes.

Example 8. *(from [Grohe, 2021]) For the graph*  *, we get:*



and its stabilizes.

2.3.3 Formal definition

Definition 9 (color refinement)**.** *Formally, we define a sequence* $(\mathsf{cr}^{(0)}(G), \mathsf{cr}^{(1)}(G), \dots)$ *of labellings as follows:*

- $\mathsf{cr}^{(0)}(G,u) = \ell_0(u)$ *for all* $u \in V$*;*
- $\mathsf{cr}^{(t+1)}(G,u) = \big(\mathsf{cr}^{(t)}(G,u), \{\!\{\mathsf{cr}^{(t)}(G,v) \mid uEv\}\!\}\big)$.

We wrote $\mathsf{cr}^{(0)}(G,u)$ instead of the cumbersome $\mathsf{cr}^{(0)}(G)(u)$. In each round, the labelling gets finer: $\mathsf{cr}^{(t+1)}(G)$ is finer than $\mathsf{cr}^{(t)}(G)$. We use the notation $\sqsubseteq$ to say 'finer than':

$$\dots \sqsubseteq \mathsf{cr}^{(2)}(G) \sqsubseteq \mathsf{cr}^{(1)}(G) \sqsubseteq \mathsf{cr}^{(0)}(G)$$

For some $t_G < |V|$, we have that $\mathsf{cr}^{(t+1)}(G)$ and $\mathsf{cr}^{(t)}(G)$ are equivalent in the following sense.

Definition 10. *Two labellings* ℓ *and* ℓ' *are equivalent if*
for all vertices $u, v \in V$, $\ell[u] = \ell[v]$ *iff* $\ell'[u] = \ell'[v]$.

main function $\mathsf{cr}(V, E, \ell_0)$
  $t := 0$
  repeat
    $\ell_{t+1} := \mathsf{refine}(V, E, \ell_t)$
    $t := t + 1$
  until ℓ_{t+1} and ℓ_t are equivalent
  return ℓ_i

function $\mathsf{refine}(V, E, \ell)$
  $\ell' :=$ new labelling
  for vertices $u \in V$ **do**
    $\ell'[u] := (\ell[u], \{\!\{\ell[v] \mid v \in E(u)\}\!\})$
  return ℓ'

Figure 2.1: Colour refinement algorithm.

Said differently, $\mathsf{cr}^{(t_G+1)}(G)$ and $\mathsf{cr}^{(t_G)}(G)$ induce the same partition. This is used as a stopping condition in colour refinement. We write $\mathsf{cr}(G)$ instead of $\mathsf{cr}^{(t_G+1)}(G)$. This is the output of colour refinement. Figure 2.1 gives the pseudo-code for colour refinement.

2.3.4 Implementation

It is possible to compute the final partition corresponding to $\mathsf{cr}(G)$ in $O((|V| + |E|) \log |V|)$.

🎓Read [Cardon and Crochemore, 1982], also [Paige and Tarjan, 1987].

The interested reader may also look at [Berkholz et al., 2017] for tight complexity.

2.3.5 Indistinguishability

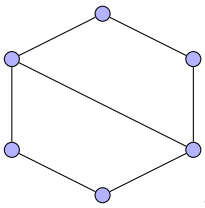
For pointed graphs, the colour at the end of colour refinement is the witness for distinguishability.

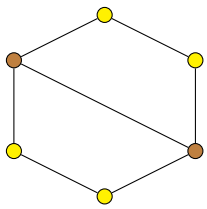
Definition 11. *G, u and G', u' are cr-indistinguishable if $\mathsf{cr}(G, u) = \mathsf{cr}(G', u')$.*

On graphs, we use the concept of *histogram*.

G, G' are cr-indistinguishable if $\mathsf{cr}(G)$ and $\mathsf{cr}(G')$ have the *same histogram of colors*: the number of vertices of a given colour are the same via $\mathsf{cr}(G)$ and $\mathsf{cr}(G')$. More formally, the histogram is defined as follows.

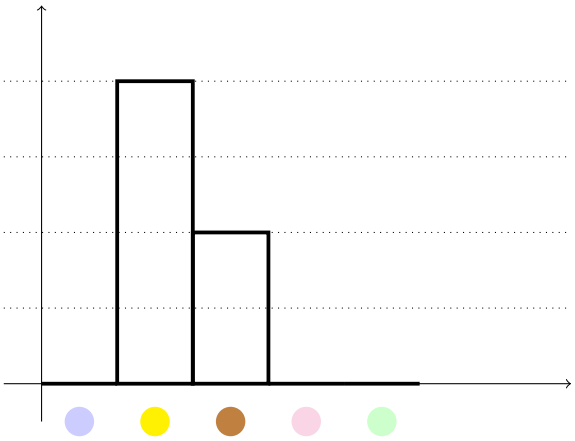
Definition 12 (histogram of G)**.** $\{\!\{\mathsf{cr}(G)\}\!\} := \{\!\{\mathsf{cr}(G, u) \mid u \in V\}\!\}$.

Example 13. *Consider the graph G:*  *. As the resulting colouring of the colour refinement procedure is*

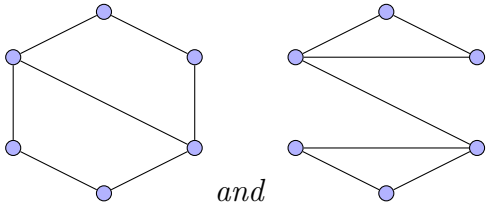


the histogram is $\{\!\{$  $\}\!\}$.

The terminology 'histogram' comes from statistics, and is a graphical representation for numbers of elements in each class of objects. We could think for a diagram like this:



Definition 14. *G, G' are cr-indistinguishable if $\{\!\{\mathsf{cr}(G)\}\!\} = \{\!\{\mathsf{cr}(G')\}\!\}$.*

Example 15. *As seen, both graphs*  *and have the same histogram* $\{\!\{$  $\}\!\}$. *Therefore, they are cr-indistinguishable.*

2.3.6 Link with isomorphism

The following proposition says that the output of cr is the same for isomorphic graphs. We say that cr is an *equivariant*.

Proposition 16. *If G and G' are isomorphic* ***then*** *G, G' are cr-indistinguable.*

Proof. Suppose that G and G' are isomorphic. Let π an isomorphism from G into G'. Given $t \in \mathbb{N}$, we consider the property $\mathcal{P}(t)$:

$$\text{for all } u \in V,\ \mathsf{cr}^{(t)}(G,u) = \mathsf{cr}^{(t)}(G',\pi(u)).$$

For $t = 0$, $\mathcal{P}(0)$ holds by definition of an isomorphism.
Suppose $\mathcal{P}(t)$ for some t. The computation is:

$$\begin{aligned}
\mathsf{cr}^{(t+1)}(G,u) &:= (\mathsf{cr}^{(t)}(G,u), \{\!\{\mathsf{cr}^{(t)}(G,v) \mid v \in E(u)\}\!\}) \\
&= (\mathsf{cr}^{(t)}(G',\pi(u)), \{\!\{\mathsf{cr}^{(t)}(G',\pi(v)) \mid v \in E(u)\}\!\}) \\
&= (\mathsf{cr}^{(t)}(G',\pi(u)), \{\!\{\mathsf{cr}^{(t)}(G',\pi(v)) \mid \pi(v) \in E'(\pi(u))\}\!\}) \\
&= (\mathsf{cr}^{(t)}(G',\pi(u)), \{\!\{\mathsf{cr}^{(t)}(G',v') \mid v' \in E'(\pi(u))\}\!\}) \\
&= \mathsf{cr}^{(t+1)}(G',\pi(u))
\end{aligned}$$

Hence $\mathcal{P}(t+1)$.
In particular:

- The condition of the repeat until loop is thus obtained for the same t in $\mathsf{cr}(G)$ and $\mathsf{cr}(G')$.
- and $\{\!\{\mathsf{cr}^{(t)}(G,u) \mid u \in V\}\!\} = \{\!\{\mathsf{cr}^{(t)}(G',\pi(u)) \mid u \in V\}\!\} = \{\!\{\mathsf{cr}^{(t)}(G',u') \mid u' \in V'\}\!\}$.

So $\{\!\{\mathsf{cr}(G)\}\!\} = \{\!\{\mathsf{cr}(G')\}\!\}$. □

The algorithm cr does not characterize isomorphism, as shown in Figure 2.2.

Proposition 17. *[Immerman and Lander, 1990] Let G and G' be two* trees*.*
G, G' are isomorphic iff G, G' are cr-indistinguable.

2.3.7 Analysis of failure

Proposition 18. *If two graphs with n vertices with the same features are d-regular[2] then they are cr-indistinguishable.*

Example 19. *Figure 2.2 shows two non isomorphic 3-regular graphs with both 20 vertices. They are not isomorphic because decaprismane has 4-cycles while dodecahedrane does not. This is sad because dodecahedrane is very stable thermically: it melts only above 4300°. On the other side, decaprismane is not very stable: its ring shape is fragile. It means that it is impossible to lean* thermal stability *by means of standard GNNs.*

Interestingly the probability that cr fails on two graphs with n vertices taken uniformly randomly goes to 0 when n goes to $+\infty$.

Theorem 20 ([Babai et al., 1980], [Arvind et al., 2015])**.** *Let G_n, G'_n be two independent uniformly[3] random graphs with n vertices. We have:*

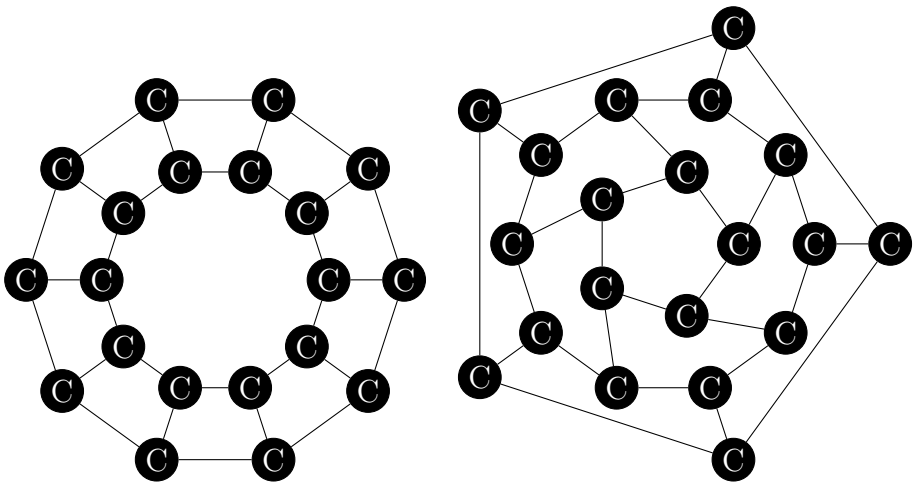
$$\mathbb{P}(G_n, G'_n \text{ are cr-indistinguable} \mid G_n, G'_n \text{ are isomorphic}) \xrightarrow[n\to+\infty]{} 1.$$

🎓Read [Babai et al., 1980], [Arvind et al., 2015] and provide a proof of the above theorem

2.4 Colour refinement and GNNs

Intuitively, GNNs are weaker than cr: cr stores in the whole 'colour' which correspond to all the arguments to compute the label of a vertex, while a GNN only stores the result. This fact is stated in Theorem VIII.1 in [Grohe, 2021], as well as [Xu et al., 2019] [Morris et al., 2019].

[2] A graph is *d-regular* if all vertices have the same degrees d.
[3] under the *Erdős–Rényi random graph model*



(a) Decaprismane (b) Dodecahedrane

Figure 2.2: Decaprismane and dodecahedrane. Two non isomorphic graphs that are 1-WL-indistinguishable [Sato, 2020].

2.4.1 Colour refinement is 'self-contained'

Colour refinement contains all the information in order to compute the output of a GNN.

Theorem 21. *Let N be a GNN with L layers. For all $t \in \{0, \dots, L\}$, for all pointed graphs G, u and G', u', we have:*

$$\mathsf{cr}^{(t)}(G, u) = \mathsf{cr}^{(t)}(G', u') \text{ then } N^{(t)}(G, u) = N^{(t)}(G', u').$$

Proof. We prove it induction on t.

□

Said, differently, the partition of $\mathsf{cr}^{(L)}(G)$ is finer than the one of $N(G)$. We make an abuse of notation and write $N(G)$ or $N^{(t)}(G)$ to denote the corresponding partition.

$$\mathsf{cr}^{(t)}(G) \sqsubseteq N^{(t)}(G).$$

In particular, we have $\mathsf{cr}(G) \sqsubseteq N^{(t)}(G)$. So:

Corollary 22. $\mathsf{cr}(G) \sqsubseteq N^{(L)}(G)$.

2.4.2 GNNs as Powerful as Colour Refinement

Is there a GNN that computes the cr-partition? In Theorem VIII.4 of [Grohe, 2021] partially answers the question. The answer is partial because the existence of a GNN is not uniform: we have one GNN for each size of graphs. We reformulate this result here.

Theorem 23. *For all integers $n \in \mathbb{N}$, there is a GNN N such that for all graphs G with at most n vertices and where the initial labellings of each vertex is in $\{0, 1\}^k$, we have*

$$N^{(n)}(G) \sqsubseteq \mathsf{cr}(G).$$

🎓Read the proof of Theorem VIII.4 in [Grohe, 2021]

There is a uniform version, see Lemma C.1 and Proposition C.2 in [Soeteman et al., 2026].

2.5 Generalizations (*)

Figure 2.2 shows two non-isomorphic regular graphs that cr is unable to distinguish. So the natural idea is to generalize 1-WL as follows: instead of just colouring vertices, we colour tuples of vertices. We introduce the two main generalizations with the notations of [Morris et al., 2023]: k-FWL = k-folkore WL, and k-OWL = k-oblivious WL. The notion of k-FWL is used in machine learning papers. OWL is a bit less powerful than FWL but it makes the bridge with colour refinement. Here are the relations of the different Weisfeiler-Leman tests:

colour refinement	2-FWL	3-FWL	...
$\equiv$ 2-OWL	$\equiv$ 3-OWL	$\equiv$ 4-OWL	...

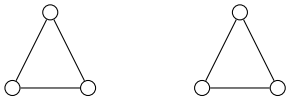
$\longrightarrow$ more and more powerful

2.5.1 k-FWL test (for $k \geq 2$)

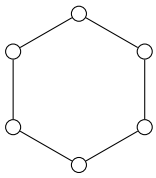
The algorithm cr has been extended on k-tuples of vertices. Given a labelled graph G, given a k-tuple $\vec{v} = (v_1, \ldots, v_k)$ of vertices, we denote by $G[\vec{v}]$ the subgraph of G induced by $\{v_1, \ldots, v_k\}$. More precisely, $G[\vec{v}] = (V', E', \ell'_0)$ is the graph whose vertices are $V' = \{1, \ldots, k\}$ and $E' = \{(i, j) \mid v_i E v_j\}$ and $\ell'_0(i) := \ell_0(v_i)$.

- Initialization: the colour of $\vec{v}$ is $G[\vec{v}]$;
- Refinement step: look at k-tuples overlapping with a tuple in all but one component. Make a *single multiset* of *tuples of values*.

Example 24. *We apply 2-OWL on the the two following graphs G and H (Figure 1 in [Huang and Villar, 2021]):*



G

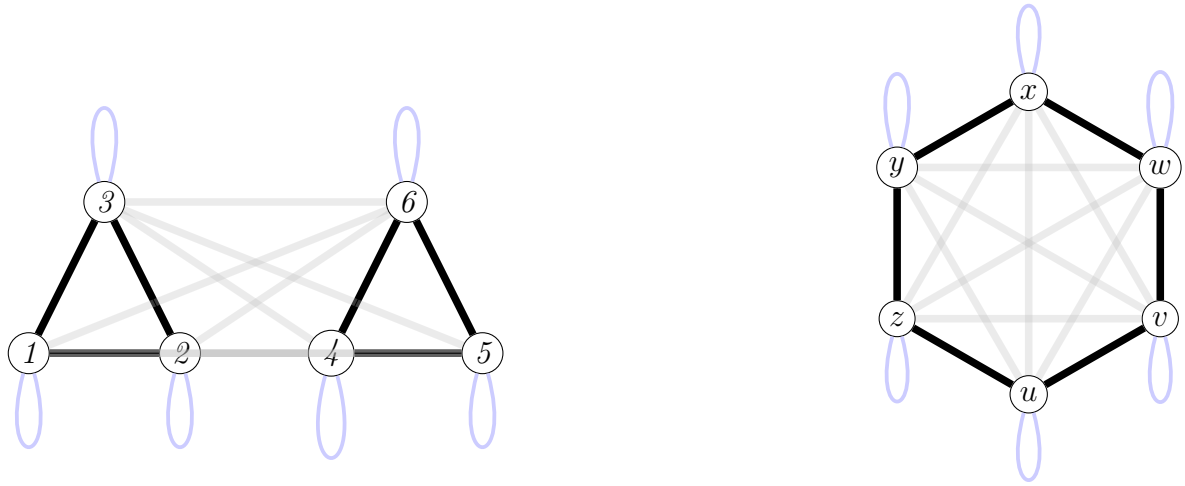


H

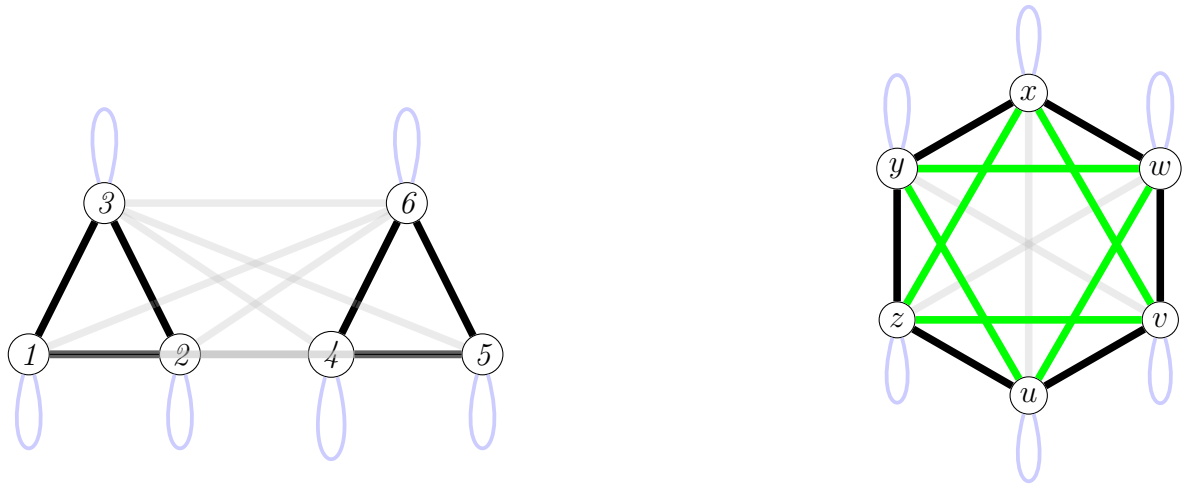
Now, let us look at all the tuples. Initially, there are three sorts of pairs:

- *Pairs (u, u) where the coordinates are equal;*
- *Pairs (u, v) with a edge between u and v;*
- *Pairs (u, v) without any edge between u and v.*

So the colours at step 0 are as follows:



Said differently, the colours of pairs exactly reflects the non-edges (gray), the edges (black) and the loops. Now, after step 1 we obtain:



The colours of pairs in G does not change (formally they change but it stabilizes). However, in graph H there are some changes. Some of the previously gray pairs differ now. Let us give the definition to see why.

Definition 25 (k-folkore WL)**.** *We define $\mathsf{fwl}_k^{(0)}, \mathsf{fwl}_k^{(1)}, \dots$ as follows:*

- $\mathsf{fwl}_k^{(0)}(G, \vec{u}) = G[\vec{v}]$;
- $\mathsf{fwl}_k^{(t+1)}(G, \vec{u}) := (\mathsf{fwl}_k^{(t)}(G, \vec{u}), \{\!\{ \left(\mathsf{fwl}_k^{(t)}(G, \vec{u}_{[1:=w]}), \dots, \mathsf{fwl}_k^{(t)}(G, \vec{u}_{[k:=w]}) \right) \mid w \in V \}\!\})$

where $\vec{u}_{[i:=w]}$ is the vector $\vec{u}$ in which the i-th coordinate has been replaced by vertex w.

Example 26. *In our example, we get:*
$\mathsf{fwl}_2^{(1)}(G, uv) := (\mathsf{fwl}_k^{(t)}(G, uv), \{\!\{ \left(\mathsf{fwl}_2^{(0)}(G, wu), \mathsf{fwl}_2^{(0)}(G, uw) \right) \mid w \in V \}\!\})$.
We have $\mathsf{fwl}_2^{(1)}(G, yw) := (\blacksquare, \{\!\{(\blacksquare, \blacksquare), (\blacksquare, \blacksquare), (\blacksquare, \blacksquare), (\blacksquare, \blacksquare), (\blacksquare, \blacksquare), (\blacksquare, \blacksquare), \}\!\})$.
But: $\mathsf{fwl}_2^{(1)}(G, yv) := (\blacksquare, \{\!\{(\blacksquare, \blacksquare), (\blacksquare, \blacksquare), (\blacksquare, \blacksquare), (\blacksquare, \blacksquare), (\blacksquare, \blacksquare), (\blacksquare, \blacksquare), \}\!\})$. *They are different: we repaint in gray and green...*

2.5.2 k-OWL test (for $k \geq 2$)

- Initialization: same as k-FWL;

- Refinement step: look at k-tuples overlapping with a tuple in all but one component. Make *tuple of multisets* of *values*.

Definition 27 (k-oblivious WL)**.** *We define* $\mathsf{owl}_k^{(0)}, \mathsf{owl}_k^{(1)}, \ldots$ *as follows:*

- $\mathsf{owl}_k^{(0)}(G, \vec{u}) = G[\vec{v}]$*;*

- $\mathsf{owl}_k^{(t+1)}(G, \vec{u}) := (\mathsf{owl}_k^{(t)}(G, \vec{u}), (\{\!\{\mathsf{owl}_k^{(t)}(\vec{u}_{[1:=w]}) \mid w \in V\}\!\}, \ldots, \{\!\{\mathsf{owl}_k^{(t)}(\vec{u}_{[k:=w]}) \mid w \in V\}\!\}))$

2.5.3 Relations

Proposition 28. *cr and 2-OWL are as powerful.*

Proof. 2-OWL can be redefined as follows.

- $\mathsf{owl}_2^{(0)}(u, v) = (\ell_0(u), \ell_0(v), 1_{uEv})$;

- $\mathsf{owl}_2^{(t+1)}(u, v) = (\mathsf{owl}_2^{(t)}(u, v), \{\!\{\mathsf{owl}_2^{(t)}(w, v) \mid w \in V\}\!\}, \{\!\{\mathsf{owl}_2^{(t)}(u, w) \mid w \in V\}\!\})$.

The goal is to prove that $\mathsf{cr}^{(t)}(u) = \mathsf{cr}^{(t)}(u')$ iff $\mathsf{owl}_2^{(t)}(u, u) = \mathsf{owl}_2^{(t)}(u', u')$. We consider a stronger induction hypothesis which is denoted by $\mathcal{P}(t)$: for all $u, v, u', v' \in V$, $\mathsf{owl}_2^{(t)}(u, v) = \mathsf{owl}_2^{(t)}(u', v')$ iff $\mathsf{cr}^{(t)}(u) = \mathsf{cr}^{(t)}(u')$ and $\mathsf{cr}^{(t)}(v) = \mathsf{cr}^{(t)}(v')$ and (uEv iff $u'Ev'$).

- $\mathcal{P}(0)$ is OK.

- Suppose $\mathcal{P}(t)$. Let us prove that $\mathcal{P}(t+1)$.

$\mathsf{owl}_2^{(t+1)}(u,v) = \mathsf{owl}_2^{(t+1)}(u',v')$

iff $\mathsf{owl}_2^{(t)}(u,v) = \mathsf{owl}_2^{(t)}(u',v')$
and $\{\!\{\mathsf{owl}_2^{(t)}(w,v) \mid w \in V\}\!\} = \{\!\{\mathsf{owl}_2^{(t)}(w,v') \mid w \in V\}\!\}$
and $\{\!\{\mathsf{owl}_2^{(t)}(u,w) \mid w \in V\}\!\} = \{\!\{\mathsf{owl}_2^{(t)}(u',w) \mid w \in V\}\!\}$
(By rewriting the multiset equalities with bijections)

iff $\mathsf{cr}^{(t)}(u) = \mathsf{cr}^{(t)}(u')$ and $\mathsf{cr}^{(t)}(v) = \mathsf{cr}^{(t)}(v')$ and $(uEv$ iff $u'Ev')$
$\exists$ bijection $\varphi : V \to V \mid \forall w \in V, \mathsf{owl}_2^{(t)}(w,v) = \mathsf{owl}_2^{(t)}(\varphi(w),v')$
$\exists$ bijection $\psi : V \to V \mid \forall w \in V, \mathsf{owl}_2^{(t)}(u,w) = \mathsf{owl}_2^{(t)}(u',\psi(w))$
(By $\mathcal{P}(t)$)

iff $\mathsf{cr}^{(t)}(u) = \mathsf{cr}^{(t)}(u')$ and $\mathsf{cr}^{(t)}(v) = \mathsf{cr}^{(t)}(v')$ and $(uEv$ iff $u'Ev')$
$\exists$ bijection $\varphi : V \to V \mid \forall w \in V,$
$\mathsf{cr}^{(t)}(w) = \mathsf{cr}^{(t)}(\varphi(w))$ and $(wEv$ iff $\varphi(w)Ev')$
$\exists$ bijection $\psi : V \to V \mid \forall w \in V,$
$\mathsf{cr}^{(t)}(w) = \mathsf{cr}^{(t)}(\psi(w))$ and $(uEw$ iff $u'E\psi(w))$
(By rewriting the bijection stuff with multiset equalities)

iff $\mathsf{cr}^{(t)}(u) = \mathsf{cr}^{(t)}(u')$ and $\mathsf{cr}^{(t)}(v) = \mathsf{cr}^{(t)}(v')$ and $(uEv$ iff $u'Ev')$
$\{\!\{\mathsf{cr}^{(t)}(w) \mid wEv\}\!\} = \{\!\{\mathsf{cr}^{(t)}(w') \mid w'Ev'\}\!\}$
and $\{\!\{\mathsf{cr}^{(t)}(w) \mid \text{ not } wEv\}\!\} = \{\!\{\mathsf{cr}^{(t)}(w') \mid \text{ not } w'Ev'\}\!\}$
$\{\!\{\mathsf{cr}^{(t)}(w) \mid uEw\}\!\} = \{\!\{\mathsf{cr}^{(t)}(w') \mid u'Ew'\}\!\}$
and $\{\!\{\mathsf{cr}^{(t)}(w) \mid \text{ not } uEw\}\!\} = \{\!\{\mathsf{cr}^{(t)}(w') \mid \text{ not } u'Ew'\}\!\}$
(By removing the equalities with the not since the union is $\{\!\{\mathsf{cr}^{(t)}(w) \mid w \in W\}\!\}$)

iff $\mathsf{cr}^{(t)}(u) = \mathsf{cr}^{(t)}(u')$ and $\mathsf{cr}^{(t)}(v) = \mathsf{cr}^{(t)}(v')$ and $(uEv$ iff $u'Ev')$
$\{\!\{\mathsf{cr}^{(t)}(w) \mid wEv\}\!\} = \{\!\{\mathsf{cr}^{(t)}(w') \mid w'Ev'\}\!\}$
$\{\!\{\mathsf{cr}^{(t)}(w) \mid uEw\}\!\} = \{\!\{\mathsf{cr}^{(t)}(w') \mid u'Ew'\}\!\}$
(By definition of $\mathsf{cr}^{(t+1)}$)

iff $\mathsf{cr}^{(t+1)}(u) = \mathsf{cr}^{(t+1)}(u')$ and $\mathsf{cr}^{(t+1)}(v) = \mathsf{cr}^{(t+1)}(v')$

So $\mathcal{P}(t+1)$.

□

More generally:

Proposition 29. *For all $k \geq 2$, k-FWL is as powerful as $(k+1)$-OWL.*

Proposition 30. *For all $k \geq 2$, $k+1$-OWL is strictly more powerful than k-OWL.*

🎓Prove the two last propositions

Further reading

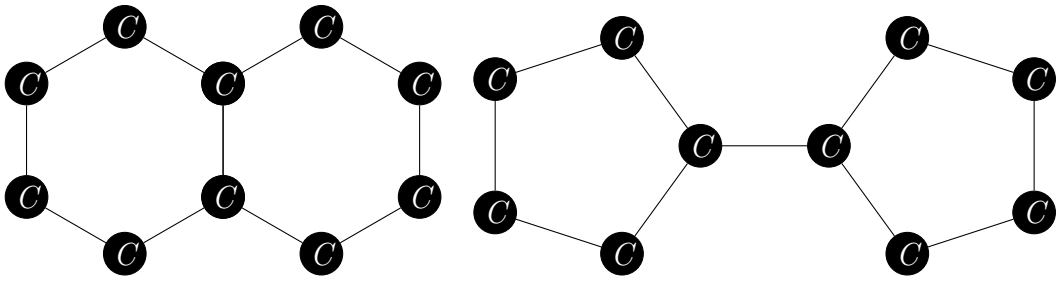
2.5.4 Higher-order GNNs

Morris et al. [Morris et al., 2019] have proposed generalization GNNs that corresponds to k-OWL for $k > 2$.

Exercises

Exercise 4. *Play with* `https://holgerdell.github.io/color-refinement/`

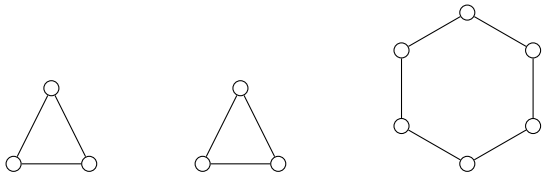
Exercise 5. *Consider these two molecules (from [Sato, 2020]):*



(a) Decalin *(b) Bicyclopentyl*

1. *Are these two graphs isomorphic?*
2. *What is the output of colour refinement?*

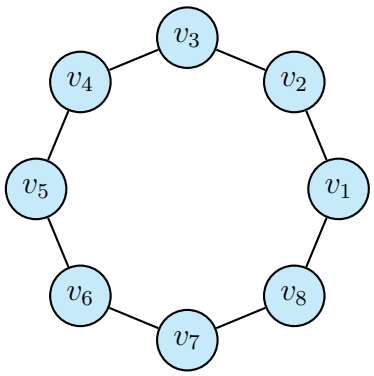
Exercise 6. *Consider the two graphs G and H (Figure 1 in [Huang and Villar, 2021]):*



G *H*

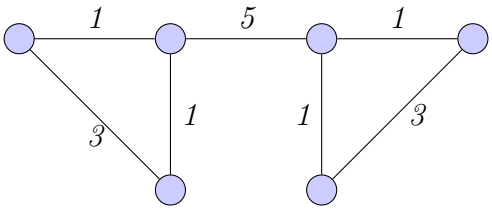
1. *Are G and H isomorphic?*
2. *Prove that color refinement does not distinguish G and H.*
3. *Prove that $2-OWL$ does not distinguish G and H.*
4. *Prove that $2-FWL$ does distinguish G and H.*

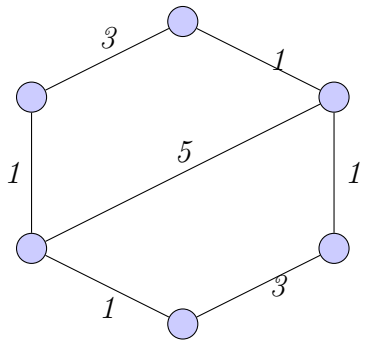
Exercise 7. *Compute 2-FWL on the following graph:*



Exercise 8 (From [Wittig et al., 2026], p. 50)**.**

1. *Give a definition of 1-WL for weighted graphs.*
2. *Prove that the two following graphs have the same histogram:*





3. *Deduce that no GNNs with weighted sums can compute the cost of a minimum spanning tree in a graph.*

Exercise 9. *Propose an efficient implementation of algorithm* cr.

Chapter 3

Logics

Key reference: [Barceló et al., 2020]

3.1 Motivation

A *property* on graphs (resp. pointed graphs) is a subset of graphs (resp. graphs). On the one hand, to any GNN N, there is a property attached to it: the subset of all graphs classified as positive. The corresponding property is written $[\![N]\!]$:

$$[\![N]\!] := \{G \mid N(G) = \top\}\,.$$

For a GNN classifying pointed graphs:

$$[\![N]\!] := \{G, u \mid N(G, u) = \top\}\,.$$

On the other hand, logic (as a science) gives languages (we call them logics!) to express properties on graphs/pointed graphs.

Example 31. *The property 'having an isolated vertex' can be expressed in first-order logic by* $\exists x \forall y \neg E(x, y)$*. We write*

$$[\![\exists x \forall y \neg E(x, y)]\!] := \{\textit{all pointed graphs where the pointed graph is isolated}\}\,.$$

In this chapter we review the basics in logic, the complexity of the satisfiability problem and some connections with colour refinement and GNNs.

What seem to be the important ingredients?

1. counting, because both GNNs and color refinement can count (ah, multisets...)

2. properties invariant via *unraveling*

3.2 First-order logic

First-order logic (FO) validity/satisfiability problem is undecidable [Turing, 1937]. More precisely:

	on finite models	on all models
satisfiability	undecidable in RE (enumerate the finite models)	undecidable in coRE
validity	undecidable in coRE	undecidable in RE (enumerate the finite proofs)

where RE means 'recursively enumerable'.

Proposition 32. *Two finite graphs are isomorphic iff they satisfy the same* FO*-formulas.*

⚠ It is false for infinite graphs/models. Two models can be non isomorphic but satisfy the same FO-formulas (see Löwenheim-Skolem theorem etc.).

3.3 First-order logic with counting

We replace the quantification $\exists x\varphi$ by $\exists^{\geq k} x\varphi$ (there are at least k elements x such that φ holds). The obtained logic is called *first-order logic with counting* (FOC).

Definition 33. *We define* $G, \lambda \models \exists^{\geq k} x\varphi$ *as follows:*

there exists $u_1, \ldots, u_k \in V$ *all distinct such that for all* $i = 1..k$*, we have* $G, \lambda[x := u_i] \models \varphi$.

We can thus define also the following macros for respectively "there are at most k elements x such that φ holds" and "there are exactly k elements x such that φ holds":

$$\exists^{\leq k}\varphi := \neg\exists^{\geq k+1} x\varphi$$
$$\exists^{=k}\varphi := \exists^{\geq k} x\varphi \wedge \exists^{\leq k}\varphi$$

Proposition 34. *FOC and FO (with equality) have the same expressivity.*

Proof. $\exists^{\geq k} x\varphi$ is rewritten in

$$\exists x_1 \ldots \exists x_k \left(\bigwedge_{i<j} x_i \neq x_j \quad \wedge \quad \bigwedge_i \varphi(x_i) \right).$$

□

3.4 Modal logic

3.4.1 Syntax

Modal logic [Blackburn et al., 2001] extends *propositional logic* with special operators $\Box$ and $\Diamond$ called *modalities*. In the standard reading, the construction $\Box\varphi$ is read as φ is necessarily true.

Definition 35 (language of basic modal logic)**.** *A modal formula is a construction generated by the following rule:*

$$\varphi ::= \bot \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Diamond\varphi$$

where p *ranges over the set of atomic propositions.*

Definition 36 (modal depth)**.** *Modal depth $md(\varphi)$ is defined by induction as follows:*

$$\begin{aligned} md(p) &= p \\ md(\neg\varphi) &= md(\varphi) \\ md(\varphi \vee \psi) &= \max(md(\varphi), md(\psi)) \\ md(\Diamond\varphi) &= 1 + md(\varphi) \end{aligned}$$

3.4.2 Semantics

Recall that a Kripke model is just a labelled graph. A *pointed Kripke model* is a pair G, u where $G = (V, E, \ell_0)$ is a Kripke model and v is a world in V.

Definition 37 (truth conditions)**.** *Given $G = (V, E, \ell_0)$, $u \in V$, $\varphi \in \mathcal{L}$ we define $G, u \models \varphi$ by structural induction over φ:*

- $G, u \not\models \bot$*;*
- $G, u \models p$ *iff* $\ell_0(p) = 1$*;*
- $G, u \models \neg\varphi$ *iff* $G, u \not\models \varphi$*;*
- $G, u \models \varphi \vee \psi$ *iff* $G, u \models \varphi$ *or* $G, u \models \psi$*;*
- $G, u \models \Diamond\varphi$ *iff there is a* $v \in E(u)$ *we have* $G, v \models \varphi$*.*

We also introduce the dual modal construction $\Box\varphi$ which is equivalent to $\neg\Box\neg\varphi$.

Example 38. *Construct a pointed graph satisfying $\Box\Diamond p$. Another one satisfying $\Diamond\Box p$.*

3.4.3 Standard translation

The standard translation consists in translating any modal formula φ into a first-order formula $\varphi'(x)$ with one single free variable.

$$\begin{aligned} tr_x(p) &::= p(x) \\ tr_x(\Diamond\varphi) &::= \exists y x E y \wedge tr_y(\varphi) \end{aligned}$$

Note that we only use two variables x and y. We write FO_2 for the fragment of FOrestricted to formulas containing only two variables. So it is interesting to note that ML is a fragment of FO_2.

3.5 Graded Modal logic

3.5.1 Definition

Graded modal logic [Tobies, 1999] is like modal logic but operator $\Diamond^{\geq k}\varphi$. Its semantics is:

$G, u \models \Diamond^{\geq k}\varphi$ there are k distinct $v_1, \ldots, v_k$ such that for all $i = 1..k$ uEv_i and $G, u_i \models \varphi$

In the same way, GML is a fragment of FOC_2, the fragment of FOC to formulas with two variables:

$$tr_x(p) ::= p(x)$$
$$tr_x(\Diamond^{\geq k}\varphi) ::= \exists^{\geq k} y xEy \wedge tr_y(\varphi)$$

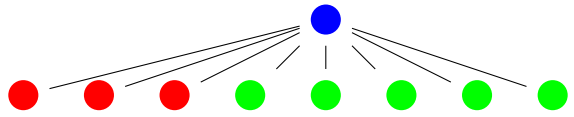
At the end, we will know how to solve the satisfiability problem of GML. But let us start to tackle the satisfiability problem of K.

3.5.2 Link with colour refinement

Each colour c of any round t can be fully described by a GML-formula.

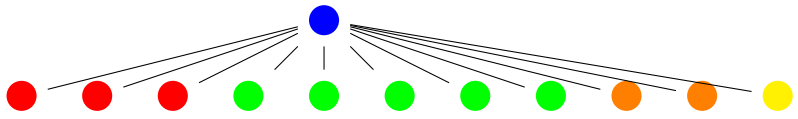
Example 39. *The colour* $(blue, \{\!\{red, red, red, green, green, green, green, green\}\!\})$ *at round 1 is captured by the* GML*-formula*

$$blue \wedge \Diamond^{=3} red \wedge \Diamond^{=5} green \wedge \Box(red \vee green)$$



where $blue, red, green$ *are formulas that capture initially colours (i.e. fully describe the labels).*

The conjunct $\Box(red \vee green)$ *says that the successors are all red or green. With it, we would just say that there are 3 red successors, and 5 green ones and nothing would have prevented having other colours. For instance:*



Theorem 40. *For all rounds* t*, for all colours* c *of round* t*, there exists a* GML *formula* $\varphi_{t,c}$ *of modal depth* t *such that*

$$\mathsf{cr}^{(t)}(G, u) = c \text{ iff } G, u \models \varphi_{t,c}.$$

Proof. By induction on t.

Base case. We take $\varphi_{0,c}$ to be a Boolean formula that describes c.

Example 41. *If* $c = \begin{pmatrix} 2 \\ -6.5 \end{pmatrix}$*, then we take* $\varphi_{0,c} = (x_1 = 2) \wedge (x_2 = -6.5)$.

Inductive case Consider the color $c = (c', M)$ where c' is a color of round $r-1$, and M is a multiset of colors of round $r-1$ too. We set:

$$\varphi_{t,c} := \varphi_{t-1,c'} \quad \wedge \quad \bigwedge_{c'' \in M} \Diamond^{=count(c'',M)} \varphi_{t-1,c''} \quad \wedge \quad \Box \bigvee_{c'' \in M} \varphi_{t-1,c''}.$$

where $count(c'', M)$ is the number of occurrences of c'' in M.

□

Now, we state that (G, u) and (G, u') are cr-indistiguishable iff they satisfy the same formulas of GML.

Theorem 42 ([Grohe, 2021], p. 6, Th. V.10)**.** *We have:*

$$\mathsf{cr}(G, u) = \mathsf{cr}(G', u') \text{ iff for all } \mathsf{GML}\text{-formulas } \varphi,\ G, u \models \varphi \text{ iff } G', u' \models \varphi.$$

Proof. (Proof given in Appendix in [Grohe, 2021])
We prove the following property $\mathcal{P}(t)$ by induction on t.

1. Color refinement gives the same results to G, u and G', u' after t rounds: $\mathsf{cr}^{(t)}(G, u) = \mathsf{cr}^{(t)}(G', u')$.

2. G, u and G', u' satisfy the same formulas φ in GML of modal depth at most t.

Base case.

Color refinement gives the same results to G, u and G', u' after 0 rounds
iff
G, u and G', u' have the same labellings
iff
G, u and G', u' satisfy the same Boolean formulas
iff
G, u and G', u' satisfy the same formulas φ in GML of modal depth at most 0.

Inductive case Suppose $\mathcal{P}(t-1)$ and prove $\mathcal{P}(t)$.

- $(1 \Rightarrow 2)$ Suppose 1. Consider a GML-formula φ. The formula φ is a Boolean combination of atoms $(x_i = 1)$ or subformulas $\Diamond^{\geq N}\psi$. First, G, u and G', u' satisfy the same atoms. By $\mathcal{P}(t-1)$, for all colours c, either all successors of u coloured by c all satisfy ψ or none of them. As both u and u' have the same number of successors of a given color, we have $G, u \models \Diamond^{\geq N}\psi$ iff $G', u' \models \Diamond^{\geq N}\psi$. To conclude, $G, u \models \varphi$ iff $G', u' \models \varphi$.

- $(2 \Rightarrow 1)$ We prove not $1 \Rightarrow$ not 2. Suppose that the colours of u and u' after t rounds are different. Then let c be the colour of u after t rounds. We have $G, u \models \varphi_{t,c}$ while $G', u' \not\models \varphi_{t,c}$ where $\varphi_{t,c}$ is given by Theorem 40. Hence not 2.

□

3.6 Link with GNNs

In the sequel, we suppose that initial features are Boolean in the following sense: for all vertices u, $\ell_i(u) \in \{0, 1\}$. Then there is a correspondence between an atomic propositional variable p_i (usual in logic) being true at vertex u and $\ell_i(u) = 1$.

Example 43. *Here is an example of a Kripke model (on the left) and its corresponding input to a GNN:*

Proposition 44 (Prop. 4.1 [Barceló et al., 2020])**.** *For all* GML*-formula φ, there is a GNN N such that for all G, u we have:*

$$G, u \models \varphi \text{ iff } N(G, u) = \top.$$

Furthermore, that the activation function of N is truncReLU*, the aggregation is sum, the combine function is a 1-layer neural network.*

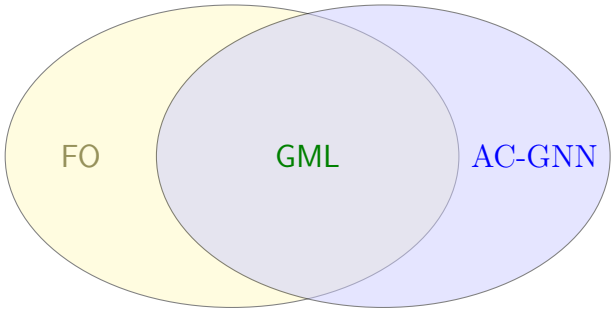
Proof. We postpone the proof to the next chapter, in which we prove a stronger result. □

Proposition 45 (Prop. 4.2 [Barceló et al., 2020])**.** *Let N be a GNN that is* FO*-expressible (with any activation/aggregation/combine function). Then there is a* GML*-formula φ such that*

$$G, u \models \varphi \text{ iff } N(G, u) = \top.$$

The two previous propositions can be summarized as follows.

Theorem 46. GML $=$ FO $\cap$ *AC-GNN.*



Theorem 47 ([Schönherr and Lutz, 2026])**.** ML $=$ *AC-GNN with max as an aggregation function.*

Theorem 48 (Prop. 5.1 [Barceló et al., 2020])**.** $\mathsf{FOC}_2 \subseteq$ *ACR-GNN.*

3.7 AC-GNN goes outside FO

Proposition 49. *There is a vertex property in AC-GNN which is not in* FO.

Proof. Consider vertex property $\mathcal{P}(G,u)$: 'G,u has strictly more red-successors than blue-successors'. We suppose that for all vertices v, we have $\ell(v) \in \{0,1\}^2$, and we model colours as follows: v is red if $\ell_1(v) = 1$ and blue if $\ell_2(v) = 1$.

$\mathcal{P}$ can be expressed by a AC-GNN. We design our AC-GNN with a single layer:

$$N^{(1)}(G,u) := \mathsf{ReLU}\left(\begin{pmatrix}1 & -1\end{pmatrix} \times \sum_{v \in E(u)} N^{(0)}(G,u)\right).$$

The number of red-successors and blue-successors are summed via the aggregation mechanism. With the multiplication with matrix $\begin{pmatrix}1 & -1\end{pmatrix}$, we get that $N^{(1)}(G,u)$ is

$$\text{number of red-successors} - \text{number of blue-successors}.$$

The classification function $cls(x) = \begin{cases} true & \text{if } x \geq 1 \\ false & \text{otherwise.} \end{cases}$

$\mathcal{P}$ cannot be expressed by a FO-formula. This requires a tool called Ehrenfeucht-Fraissé games, see next chapter!

□

As we will see for verification purposes, it is sad to restrict ourselves to GNN that are FO-expressible.

Going further

In [Schönherr and Lutz, 2026], they also give expressivity results in the non-uniform setting: they consider the expressivity for each size of graphs.

Exercises

Exercise 10. *Give a* FOC_2*-formula* $\varphi(x)$ *for which there is no AC-GNN* N *such that* $G, u \models \varphi(x)$ *iff* $N(G, u) = \top$.

Exercise 11. *[see Example 3 in [Pratt-Hartmann, 2009]] Consider formula* $\varphi := \forall x \exists y s(x, y) \wedge \forall x \exists^{\leq 1} y s(y, x) \wedge \exists \forall y \neg s(y, x)$.

- *Prove that* φ *is satisfiable.*
- *Prove that* φ *has no finite model.*

Exercise 12. *Prove that* ML *has the expressivity than AC-GNN where the aggregation function is MAX instead of SUM. See* `https://arxiv.org/abs/2507.18145`

Exercise 13. *We define the relation* $\sim_{\#}$ *"graded bisimilation" defined in* `https://www2.mathematik.tu-darmstadt.de/~otto/papers/cml19.pdf`.

Show that $G, u \sim_{\#} G', u'$ *iff for all integers* L*, the unravellings up to level* L *of* G, u *and* G', u' *are isomorphic.*

Exercise 14. *In this exercise, we will prove that any AC-GNN that is* FO*-definable is captured by a* GML*-formula.*

1. *Show that if for all integers* L*, the unravellings up to level* L*, of* G, u *and* G', u' *are isomorphic, then for all GNNs* N*, we have* $N(G, u) = N(G', u')$.
2. *Read* `https://www2.mathematik.tu-darmstadt.de/~otto/papers/cml19.pdf` *that shows that the fragment of unary* FO *that only depend on the unravelling is* GML.
3. *Conclude.*

Chapter 4

Ehrenfeucht-Fraïssé games and pebble games

References:

- `https://pi.math.cornell.edu/~mec/Summer2009/Raluca/index.html` 6 lessons on Ehrenfeucht-Fraïssé games
- [Immerman and Lander, 1990], [Cai et al., 1992]

Ehrenfeucht-Fraissé games (EF games) ([Fraïssé, 1954], [Ehrenfeucht, 1961]) form a tool to prove that FO cannot express some property. The playground is a pair of graphs (G, G'). We have two players: Spoiler🕵 and Duplicator👤. The role of Spoiler🕵 is to prove that G and G' are different wrt to a given logic; meaning proving that there is a formula that distinguish G from G' (the formula is true in G and false in G', or the contrary). The role *Duplicator* is the contrary: to prove that the two graphs are the same wrt to some logic.

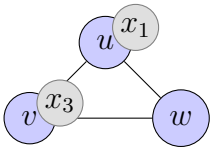
EF games are useful to prove that $\#p \geq \#q$ is not expressible in FO. EF games are generalized by pebble games that takes into account the number of variables. Pebble games are then also generalized to logics with counting. They are used to show the equivalence between colour refinement and FOC_2.

4.1 Pebbles

4.1.1 Why?

Here are examples of pebbles:

A placement of some pebbles in a given graph corresponds to an *assignment* σ of the used variables. Pebbles give a nice way to represent visually assignments.

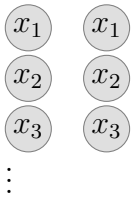
Example 50. *The following placement* 

corresponds to the assignment which the following partial mapping:

$$\begin{array}{rccc} \sigma: & Variables & \to & V \\ & x_1 & \mapsto & u \\ & x_3 & \mapsto & v \end{array}$$

4.1.2 Pair of pebbles

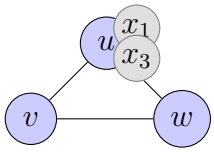
In EF games, and so-called pebble games, the playground is a pair of graphs (G, G'). That is why, we use pairs of pebbles. Each pair corresponds to a variable. Said differently, for each variable x_i, there is a pair of pebbles.

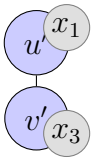


Some pairs of pebbles are unused. In that case, they are not placed and are outside the structures. When a pair  is used, then one of the pebble  is on some vertex of graph G, while the other pebble  is on some vertex of graph G'.

A placement of pebbles corresponds to two assignments with the same domain. But interestingly, it corresponds to a relation in $V \times V'$:

$$\{(u, v') \in V \times V' \mid \text{there are } x_i\text{-pebbles placed on } u \text{ and } v'\}.$$

Example 51. 



is the relation $\{(u, u'), (u, v')\}$.

4.1.3 Winning condition

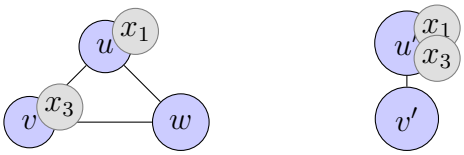
Looking at this relation will tell you whether the Spoiler was able to differentiate the structure: when the relation is non a partial isomorphism.

Definition 52. *A partial relation $R \subseteq V \times V'$ is a partial isomorphism iff:*

- *it is a bijection from its domain to its image;*
- *for all u in the domain of R, u and $R(u)$ have the same label;*
- *for all u and v in the domain, uEv iff $R(u)E'R(v)$.*

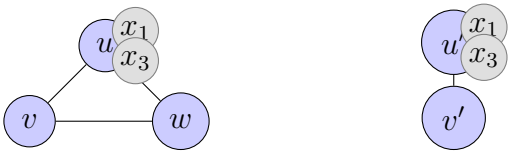
Not a partial isomorphism

Here are examples when the relation is not a partial isomorphism:

Example 53. 

is the relation $\{(u, u'), (v, u')\}$ *which is not a partial isomorphism.*

Partial isomorphism

Example 54. 

is the relation $\{(u, u')\}$ *which is a partial isomorphism.*

4.2 Ehrenfeucht-Fraïssé games

4.2.1 Description of the game

There are two players: Spoiler and Duplicator.

EF game with r rounds on G and G':
- **for** $i = 1$ to r **do**
  - Spoiler chooses a vertex in some graph, either in G or G', and put a pebble x_i on it
  - Duplicator chooses a vertex in the other graph and puts the second pebble x_i
- **if** the current relation is a partial isomorphism **then**
  - Duplicator wins
- **else**
  - Spoiler wins

Duplicator tries to prove that the two structures cannot distinguish in FO. On the contrary, when Spoiler wins, it means that Spoiler has been able to pinpoints the important vertices in the structures that make the difference.

4.2.2 Methodology

Definition 55. *The* quantifier-depth *of a* FO*-formula* φ *is the maximum number of nested quantifiers in* φ*.*

Theorem 56. *The following two statements are equivalent.*

1. *G and G' agree on* FO*-formulas of quantifier-depth at most r*
2. *Duplicator has a winning strategy for the r-round EF game on G and G'.*

Theorem 57. *A property $\mathcal{P}$ is not expressible in* FO *if there exists a infinite sequence of pair of graphs $(G_0, G'_0), (G_1, G'_1), (G_2, G'_2), \ldots$ such that for all $r \in \mathbb{N}$,*

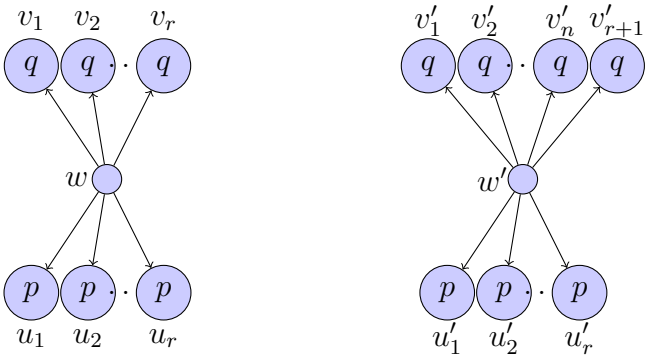
- *G_r and G'_r agree on* FO*-formulas of quantifier-depth at most r, and*
- *$G_r \models \mathcal{P}$ and $G'_r \not\models \mathcal{P}$.*

4.2.3 Application

Theorem 58. *$\#p \geq \#q$ is not expressible by a* FO *formula $\varphi(x)$.*

Proof. We show that the formula $\#p \geq \#q$ is not expressible by a FO formula $\varphi(x)$. We observe that if the property "for all vertices of a graph $\#p \geq \#q$" is not expressible in FO, then the FO formula $\varphi(x)$ doesn't exist, because if it existed the property would be expressible in FO by the formula $\forall x \varphi(x)$.

For each integer $r > 0$, we consider the graphs G_r and G'_r such that every vertices of G_r verify $\#p \geq \#q$ while this is not the case for G'_r :



G_r G'_r

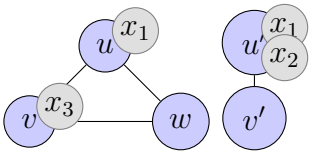
On the graphs G_r and G'_r, the duplicator wins the Ehrenfeucht-Fraïssé game with r rounds: if the spoiler chooses w (resp. w') the duplicator chooses w' (resp. w), if the spoiler chooses some u_i or v_i (resp. u'_i or v'_i) the duplicator chooses u'_j or v'_j (resp. u'_j or v'_j). (If the world chosen by spoiler has not been chosen in the previous round the duplicator pick a fresh index j. Otherwise, the duplicator picks the j corresponding to the world chosen in the previous rounds). Since there are only n distinct values that i can take, the duplicator will win the game in n rounds with her strategy. Thus the property "for every vertices of a graph, $\#p \geq \#q$" is not expressible in FO.

□

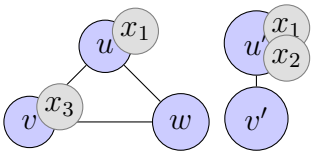
4.3 Pebble games

4.3.1 Game for first-order logic with k variables

We give the pseudo-code of pebble games for FO_k. It takes as an input the playground (the two graphs G and G'). It also takes as an input σ and σ' that are assignment in respectively G and G'. The two assignments correspond to a game configuration where some pebbles are already placed. So both σ and σ' should have the same domain.

Example 59. 

is not a suitable input σ and σ' because the domains are not the same: the domain of σ is $\{x_1, x_3\}$ while the domain of σ' is $\{x_1, x_2\}$.

Example 60. 

is a suitable input σ and σ' because the domains are the same: the domain of both σ and σ' is $\{x_1, x_3\}$.

// σ (resp. σ') is a partial assignment in G (in G') with the same domain
game $pebbleGame(G, \sigma, G', \sigma', nbVariables = k, rounds = r)$
  Put the pebbles according to σ and σ'
  for $j = 1$ to r **do**
    - Spoiler chooses a variable index $i \in \{1, \ldots, k\}$;
    - Spoiler chooses a vertex in G or G' and put the x_i-pebble on that vertex;
    - Duplicator must then place the other x_i-pebble on a vertex of the other graph
    - If the relation is not a partial isomorphism Spoiler wins
  Duplicator wins

Spoiler wins the game if Spoiler wins at some round. Duplicator wins the game if Spoiler does not win the game, i.e. if Spoiler has never been able to 'prove' that the structures are different.

Theorem 61. *We have:*

$$\begin{gathered} \textit{Duplicator has a winning strategy in} \\ pebbleGame(G, \sigma, G', \sigma', nbVariables = k, rounds = +\infty) \\ \textit{iff} \\ G, \sigma \textit{ and } G', \sigma' \textit{ satisfy the same } \mathsf{FO}_k\textit{-formulas.} \end{gathered}$$

Theorem 62. *We have:*

$$\begin{gathered} \textit{Duplicator has a winning strategy in} \\ pebbleGame(G, \sigma, G', \sigma', nbVariables = k, rounds = r) \\ \textit{iff} \\ G, \sigma \textit{ and } G', \sigma' \textit{ satisfy the same } \mathsf{FO}_k\textit{-formulas with quantifer-depth at most } r. \end{gathered}$$

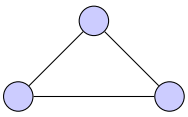
Formulas	Game
number k of variables	= number k of pairs of pebbles
quantifier-depth r	number r of rounds

4.3.2 Application

The property 'Every vertex is of degree ≥ 2' is expressed in FOC_2 by $\forall x \exists^{\geq 2} y E(x,y)$. Recall that $\forall x \varphi(x)$ is $\neg \exists^{\geq 1} x \neg \varphi(x)$. However that property cannot be said without counting.

Proposition 63. *The property 'Every vertex is of degree ≥ 2' is not expressible in FO_2.*

Proof. We consider the clique of size 2 and 3, written K_2 and K_3 respectively.



In the pebble games with 2 pairs of pebbles, whatever the spoiler is choosing, the duplicator can also reply keep the current mapping a partial isomorphism. So the duplicator has a winning strategy. So both K_2 and K_3 satisfy the same FO_2-formula. But K_2 does not satisfy the property but K_3 does. □

Corollary 64. *In terms of expressivity,* $\mathsf{FO}_2 \subsetneq \mathsf{FOC}_2$.

4.4 Pebble games with counting

Now we tackle counting. We do it restricted to k variables because we want to study the correspondence of FOC_2 and colour refinement. One could also generalize the simpler EF game framework instead.

4.4.1 Game for first-order logic with k-variables and counting

The new step is the choosing of a set A in one the graphs by Spoiler🕵; and Duplicator👤 must reply with a set B of the same size in the other graph.

game $pebbleGame'(G, \sigma, G', \sigma', nbVariables = k, rounds = r)$
- Put the pebbles according to σ and σ'
- **for** $j = 1$ to r **do**
  - Spoiler🕵 chooses a variable index $i \in \{1, \ldots, k\}$;
  - Spoiler🕵 chooses a set A of vertices from one of the graphs;
  - Duplicator👤 chooses with a set B of vertices from the other graphs
  - We should have $|B| = |A|$ otherwise Spoiler🕵 wins
  - Spoiler🕵 places one of the x_i-pebble on a vertex in B;
  - Duplicator👤 must then place the other x_i-pebble on a vertex in A.
  - If the relation is not a partial isomorphism, Spoiler🕵 wins
- Duplicator👤 wins

Example 65. *We consider the clique of size 2 and 3, written K_2 and K_3 respectively.*

Spoiler has a trivial winning strategy: choose A to be the set of all 3 vertices of K_3. *Duplicator* is unable to pick a set of size 3 in K_2! So *Spoiler* wins.

Theorem 66. *We have:*

$$\begin{gathered}\textit{Duplicator has a winning strategy in}\\ pebbleGame'(G,\sigma,G',\sigma',nbVariables=k,rounds=+\infty)\\ \textit{iff}\\ G,\sigma \textit{ and } G',\sigma' \textit{ satisfy the same } \mathsf{FOC}_k\textit{-formulas.}\end{gathered}$$

Theorem 67. *We have:*

$$\begin{gathered}\textit{Duplicator has a winning strategy in}\\ pebbleGame'(G,\sigma,G',\sigma',nbVariables=k,rounds=r)\\ \textit{iff}\\ G,\sigma \textit{ and } G',\sigma' \textit{ satisfy the same } \mathsf{FOC}_k\textit{-formulas and quantifer-depth at most } r.\end{gathered}$$

For simplicity, we focus on the case $k = 1$. Before given the big characterisation, we give the characterisation on 1-variable formulas of FOC_2. For that, we need a version of colour refinement that also consider non-neighbours.

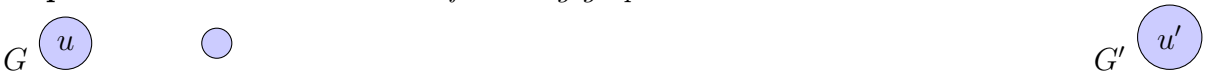
Definition 68. *The* colour refinement *algorithm that also considers non-neighbours is given by:*

- $\overline{\mathsf{cr}}^{(0)}(G,u) = \ell_0(u)$ *for all* $u \in V$*;*
- $\overline{\mathsf{cr}}^{(t+1)}(G,u) = \big(\overline{\mathsf{cr}}^{(t)}(G,u), \{\!\{\overline{\mathsf{cr}}^{(t)}(G,v) \mid (u,v) \in E\}\!\}, \{\!\{\overline{\mathsf{cr}}^{(t)}(G,v) \mid (u,v) \notin E\}\!\}\big)$.

Again $\overline{\mathsf{cr}}$ *gives the colour when stabilization.*

Note that the characterisation does not work for cr.

Example 69. *We consider the two following graphs:*



In the colour refinement cr*, all the two vertices in* G *and the vertex in* G' *have the same final colour. But* $G,u \models \exists^{\geq 2}\top$ *whereas* $G',u' \not\models \exists^{\geq 2}\top$.

With $\overline{\mathsf{cr}}$*, the colours are given by:*

Theorem 70 ([Cai et al., 1992], p. 13, Th. 5.2)**.** *Here,* $\overline{\mathsf{cr}}$ *is colour refinement but we also consider non-neighbours. We have:*

$\overline{\mathsf{cr}}(G,u) = \overline{\mathsf{cr}}(G',u')$ *iff* $G,[x_1 := u]$ *and* $G',[x_1 := u']$ *satisfy the same 1-variable* FOC_2*-formulas.*

Proof. We prove the following property $\mathcal{P}(r)$ by induction on r.

1. $\overline{\mathsf{cr}}^{(r)}(G,u) = \overline{\mathsf{cr}}^{(r)}(G',u')$;

2. $G, [x_1 := u]$ and $G', [x_1 := u']$ satisfy the same formulas $\varphi(x_1)$ in FOC_2 of quantifier depth at most r;

3. Duplicator has a winning strategy after r rounds on the FOC_2-game on $G, [x_1 := u]$ and $G', [x_1 := u']$

Base case.

$$\overline{\mathsf{cr}}^{(0)}(G, u) = \overline{\mathsf{cr}}^{(0)}(G', u')$$
iff
u and v have the same labels
iff
u and v satisfy the same Boolean formulas
iff
u and v satisfy the same same formulas $\varphi(x_1)$ in FOC_2 of quantifier depth 0
iff
initially the following mapping is an isomorphism:

$$\begin{array}{ccc} V_G & \to & V_H \\ \text{vertex in } V_G \text{ under a } x_i\text{-pebble} & \mapsto & \text{vertex in } V_H \text{ under a } x_i\text{-pebble} \end{array}$$

iff
Duplicator wins to $pebbleGame'(G, u, G', u', nbVariables = 2, rounds = 0)$

Hence $\mathcal{P}(0)$.
Inductive case. Let $r \geq 1$. Suppose $\mathcal{P}(r-1)$. Prove $\mathcal{P}(r)$.

- (not 1 $\Rightarrow$ not 2). The idea: if $\overline{\mathsf{cr}}^{(r)}(G, u) \neq \overline{\mathsf{cr}}^{(r)}(G', u')$ then we build a formula relying on the fact that GML can talk about colours!

  Either $\overline{\mathsf{cr}}$ gives already different results to G, u and G', u' after $r-1$ rounds. By induction there is a formula $\varphi(x_1)$... distinguishing them.

  Or the difference is in the last round of 1-WL. It means that u and v have not the same number of successors (let say N and N' with $N < N'$) for some color c. Let $\varphi_{c,r-1}$ be a GML-formula (GML with a modal operator for neighbors, and only for non-neighbors) that characterizes exactly the fact that the color is c with $r-1$ quantifiers (see Theorem 40). The formula $\exists^{\geq N'} y(E(x_1, y) \wedge \varphi_{c,r-1}(y)$ distinguishes u and v.

- (not 2 $\Rightarrow$ not 3). Suppose there is a formula φ of quantifier depth at most r such that $G, u \models \varphi(x_1)$ and $G', u' \not\models \varphi(x_1)$. W.l.o.g. φ is of the form $\exists^{\geq N} y \psi(x_1, y)$ (indeed, if φ is a conjunction it differs with a conjunct, etc.).

  We build a winning strategy for Spoiler. First Spoiler chooses a set A in G of size N containing vertices satisfying $\psi(y)$. More formally, for all $a \in A$, $G, [x_1 := u, y := a] \models \psi(x_1, y)$.

  As $G', u' \not\models \varphi(x_1)$, for all sets B of size N of G', there is a vertex v' such that $\psi(x_1, y)$ does not hold. More formally, $G', [x_1 := u', y := v] \not\models \psi(x_1, y)$. So whatever the choice of B is, Spoiler will place the y-pebble on v'. And boom... Duplicator must respond with some $a \in A$. But $G, [x_1 := u, y := a] \models \psi(x_1, y)$.

ke it more
cise

Now, either $G', [x_1 := u', y := v]$ and $G, [x_1 := u, y := a]$ is not a local isomorphism. In that case, Spoiler wins trivially.

Or $G', [x_1 := u', y := v]$ and $G, [x_1 := u, y := a]$ is a local isomorphism. Now let us look at the content of $\psi(x_1, y)$. The truth of $\psi(x_1, y)$ differs on $G', [x_1 := u', y := v]$ and $G, [x_1 := u, y := a]$. But then, it differs because of the truth of a subformula of the form $\exists^{\geq N} x_i \chi$ of ψ. But then, $\exists^{\geq N} x_i \chi$ is a formula with only a single free variable and of rank $r - 1$. Let say x_1 is the free variable. So, we can apply the induction hypothesis. And *Spoiler* has a winning strategy in $G', x_1 := u'$ and $G, x_1 := u$.

- $(1 \Rightarrow 3)$ Suppose that $\overline{\mathsf{cr}}^{(r)}(G, u) = \overline{\mathsf{cr}}^{(r)}(G', u')$. We have, for all colours c of rank $r - 1$, u has the same number of c-neighbours than u'. Also, u has the same number of c-non-neighbours than u'. Spoiler chooses a variable index i. Spoiler chooses a set A (w.l.o.g. in G').

  How should Duplicator choose set B? For all colours c of rang $r - 1$, Duplicator puts into B the same number of c-neighbours of u' than the number of c-neighbours of u in A. For all colours c of rang $r - 1$, Duplicator puts into B the same number of c-non-neighbours of u' than the number of c-non-neighbours of u in A. By definition, B has the same cardinality than A.

  Now Spoiler chooses some v' in B. Which vertex v should Duplicator choose in A? Just a vertex v such that $(u, v) \in E$ iff $(u', v') \in E'$, and v and v' have the same colour of rank $r - 1$. In other words, $\overline{\mathsf{cr}}^{(r-1)}(G, v) = \overline{\mathsf{cr}}^{(r-1)}(G', v')$. By induction, the duplicator has a winning strategy in the rest of the game on G, v and G', v'.

□

In [[Immerman and Lander, 1990], p. 17, Th. 1.8.1], a similar result is proven for standard colour refinement but in the same graph G (i.e. $G = G'$). This means that having the same red successors implies having the same red non-successors. However, if we consider histograms, this will also be implied. So we can get the characterisation between cr-indistinguishability and FOC_2, see Corollary 73. First we give the intermediate steps as propositions.

Proposition 71. *Consider $\overline{\mathsf{cr}}$ with neighbours and non-neighbours.*
Two graphs are $\overline{\mathsf{cr}}$-indistinguishable iff they satisfy the same closed formulas of FOC_2.

Proof. $\boxed{\Rightarrow}$ By contradiction. Suppose that there exists a formula φ true in G but not in G'. Formula φ is closed one of the conjunct is of the form $\exists^{\geq N} x \psi(x)$ is true in G and false in G'.

There are more than N vertices in G satisfying $\psi(x)$, but strictly less than N vertices in G' satisfying $\psi(x)$.

But G and G' have the same histogram. So by Theorem 70, the number of vertices satisfying $\psi(x)$ in G and the number of vertices satisfying $\psi(x)$ in G' are equal. Contradiction.

$\boxed{\Leftarrow}$ Consider the histogram of G. We can define a FOC_2-formula φ that says 'the histogram of the current graph is the one of G'. Formula φ is true on G but also on G'. So G' has the same histogram than G'. □

Proposition 72. *Two graphs are $\overline{\mathsf{cr}}$-indistinguishable iff they are cr-indistinguishable.*

Proof. $\boxed{\Rightarrow}$ Because we just have more information in $\overline{\mathsf{cr}}$ than in cr.

$\boxed{\Leftarrow}$ Suppose that the two graphs are cr-indistinguishable. It means that for all cr-colours c (at any rounds), the number of c-vertices in G is equal to the number of c-vertices in G'. Therefore, suppose $\mathsf{cr}^{(t)}(G,u) = \mathsf{cr}^{(t)}(G',u')$. It means that, for all colours c' at round $t-1$, the number of u,c'-neighbours at round $t-1$ is equal to the number of u',c'-neighbours at round $t-1$. But as they are the same number of c'-vertices in both G and G', we have the same number of u,c'-non-neighbours than the number of u',c'-non-neighbours. So $\overline{\mathsf{cr}}^{(t)}(G,u) = \overline{\mathsf{cr}}^{(t)}(G',u')$. So G and G' also have the same $\overline{\mathsf{cr}}$-histogram. □

Corollary 73. *The following statements are equivalent:*

1. *G and G' are $\overline{\mathsf{cr}}$-indistinguishable;*
2. *G and G' are cr-indistinguishable;*
3. *G and G' satisfy the same closed FOC_2-formulas.*

The general case is as follows. The proofs are similar but more cumbersome.

Theorem 74. *For all $k = 2, 3, \ldots,$*

1. *two graphs are k-FWL-indistinguishable*
2. *they satisfy the same closed formulas of FOC_{k+1}.*

As k-FWL $= k+1$-OWL, the result is neater to be phrased and also captured colour refinement by 2-OWL.

Theorem 75. *For all $k = 2, 3, \ldots,$*

1. *two graphs are k-OWL-indistinguishable*
2. *they satisfy the same closed formulas of FOC_k.*

Chapter 5

Verifying GNNs

Key references: [Nunn et al., 2024] and [Benedikt et al., 2024]

Our goal is to be able to verify GNNs as follows. We consider formulas in some logic evaluated on pointed graphs (for instance, modal logic or graded modal logic). Given a formula φ, we write $[\![\varphi]\!]$ for the set of pointed graphs (G, u) satisfying φ. Let us list some verification tasks.

1. Satisfiability problem of a GNN: Given a GNN N, is there an input (G, E, ℓ) that is positively classified by N? ($[\![N]\!] \neq \emptyset$)
2. Given a GNN N, given a specification formula φ, are the inputs positively classified by N exactly the inputs satisfying φ? ($[\![N]\!] = [\![\varphi]\!]$)
3. Given a GNN N, given a specification formula φ, are the inputs positively classified by N satisfying φ? ($[\![N]\!] \subseteq [\![\varphi]\!]$)
4. Given a GNN N, given a specification formula φ, are the inputs satisfying φ classified positively by N? ($[\![\varphi]\!] \subseteq [\![N]\!]$)
5. Given a GNN N, given a specification formula φ, does there exist an input satisfying φ and classified positively by N? ($[\![\varphi]\!] \cap [\![N]\!] \neq \emptyset$)

Link with Hoare logic

Problem 4 corresponds to the following Hoare logic triplet:

$$\{\varphi\}\, N\, \{output = true\}$$

The methodology is to design a "superlogic" in which φ as well as the computation of the GNN N can be described.

5.1 Representing a GNN with a "logic"

In this section, we introduce a *lingua franca* to describe GNNs. The language just mimics the computation performed by a GNN. We call our language AC-GNN-L. The language is inspired from the one in [Sälzer et al., 2025]. A similar language was introduced in

[Barceló et al., 2026] and is called MPLang (for message-passing language). The difference is that they operate on vectors of features (which is nicer for studying ReLU) but more combersome to make the link with modal logic and Presburger arithmetics.

5.1.1 Syntax of AC-GNN-L

We consider expressions generated by the following grammar:

$$\vartheta ::= c \mid \ell_i \mid \alpha(\vartheta) \mid agg(\vartheta) \mid \vartheta + \vartheta \mid c \times \vartheta$$

where c is any number, ℓ_i is any feature, α is any symbol to denote any activation function, agg is a symbol to represent any aggregation function (but it will interpreted as the sum in our case).

5.1.2 Semantics of AC-GNN-L

The semantics mimics the computation performed by a GNN:

$$\begin{aligned}
[\![c]\!]_{G,u} &= c, \\
[\![\ell_i]\!]_{G,u} &= \ell_i(u) \\
[\![\vartheta + \vartheta']\!]_{G,u} &= [\![\vartheta]\!]_{G,u} + [\![\vartheta']\!]_{G,u}, \\
[\![c \times \vartheta]\!]_{G,u} &= c \times [\![\vartheta]\!]_{G,u}, \\
[\![\alpha(\vartheta)]\!]_{G,u} &= [\![\alpha]\!]([\![\vartheta]\!]_{G,u}), \\
[\![agg(\vartheta)]\!]_{G,u} &= \Sigma_{v|uEv}[\![\vartheta]\!]_{G,v},
\end{aligned}$$

We write $[\![\vartheta \geq 1]\!] = \{G, u \mid [\![\vartheta]\!]_{G,u} \geq 1\}$.

5.1.3 Correspondence with GNNs

Proposition 76. *Given a GNN N, there exists an expression ϑ such that $[\![N]\!] = [\![\vartheta \geq 1]\!]$.*

Proof. See [Sälzer et al., 2025] for a formal proof (even in that paper it is given for quantized GNNs). □

Here is an example on how to translate a GNN into an expression.

Example 77. *Consider a GNN N:*

- $N^{(0)}(G, u) = \ell_0(u)$;
- $N^{(t+1)}(G, u) = \vec{\alpha}(A_t \times N^{(t)}(G, u) + B_t \times \sum\{\!\{N^{(t)}(G, v) \mid v \in E(u)\}\!\} + b_t)$ *for all* $u \in V$.

with the stopping condition $w^t \ell_L(u) + b \geq 0$.

The idea is to write the expression ϑ. To make it clearer we explain the proof via an example. Suppose:

- $N^{(0)}(G, u) = \ell_0(G, u)$;

- $N^{(1)}(G,u) = \vec{\alpha}\left(\begin{pmatrix}2 & 1\\ -1 & 4\end{pmatrix} \times N^{(0)}(G,u) + \begin{pmatrix}5 & 3\\ 2 & 6\end{pmatrix} \times \sum\{\!\{N^{(0)}(G,v) \mid v \in E(u)\}\!\} + \begin{pmatrix}1\\ -2\end{pmatrix}\right)$ *for all* $u \in V$.
- $N^{(2)}(G,u) = \vec{\alpha}\left(\begin{pmatrix}3 & 0\\ -2 & 0\end{pmatrix} \times N^{(1)}(G,u) + \begin{pmatrix}-1 & 0\\ 0 & 5\end{pmatrix} \times \sum\{\!\{N^{(1)}(G,v) \mid v \in E(u)\}\!\} + \begin{pmatrix}0\\ 0\end{pmatrix}\right)$ *for all* $u \in V$.

We suppose the stopping condition to $2N^{(2)}(G,v)[1] + 3N^{(2)}(G,v)[2] \geq 1$.

Then, the corresponding AC-GNN-L expression ϑ_N is given by:

$$\begin{aligned}
\psi_1 &= \alpha(2\ell_1 + \ell_2 + 5agg(\ell_1) - 3agg(\ell_2) + 1),\\
\psi_2 &:= \alpha(-\ell_1 + 4\ell_2 + 2agg(\ell_1) + 6agg(\ell_2) - 2),\\
\chi_1 &:= \alpha(3\psi_1 - agg(\psi_1),\\
\chi_2 &:= \alpha(-2\psi_1 + 5(agg(\psi_2)),\\
\varphi_A &:= 2(\chi_1) - \chi_2 \geq 1.
\end{aligned}$$

Note that some expressions do not represent a GNN with a single layer neural network as an combination function. For instance, $agg(2 \times x)$ does not syntactically correspond to a GNN. Indeed, aggregation is always computed on values of the form $\alpha(...)$ because it aggregates over values of the previous layer. Also a GNN is represented by a linear expression over $\alpha(...)$. We say that these kind of expressions are *nice*.

Proposition 78. *For all nice AC-GNN-expressions* ϑ*, there is a AC-GNN* N *such that* $[\![\vartheta \geq 1]\!] = [\![N]\!]$.

In [Barceló et al., 2026], they study the general language of AC-GNN-expressions. Said differently, they allow for *linear layers without an activation function.*

5.2 Logic $K^{\#}$

We now define logic $K^{\#}$ defined in [Nunn and Schwarzentruber, 2023] and [Nunn et al., 2024]. It can be thought as modal logic with Presburger arithmetics: modal logic where you can count the number of successors, compare the numbers, but also perform addition and multiplication.

Example 79. *More red successors than blue ones is expressed by the* $K^{\#}$*-formula*

$$\#red \geq \#blue.$$

Example 80. $dog \rightarrow (\#leg = 4)$*: if the current vertex is a dog, then the number of successors that are legs is equal to 4.*

Example 81. $\#((\#fingers) = 5) = 4$*: there are 4 successors that have 5 successors that are fingers.*

Example 82. $\#hydrogen = 2\#carbon$*: the number of successors that are hydrogen is two times the number of successors that are carbon.*

A similar logic is defined in [Benedikt et al., 2024] but it does not have the 1_φ construction. Also they present the language as a sort guarded fragment of FO (even though the language is more expressive than FO!). It has the same expressivity but probably not the same succinctness.

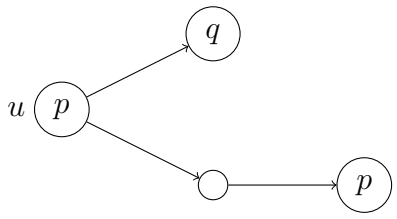


Figure 5.1: Example of a pointed graph G, u. We indicate true propositional variables at each vertex.

5.2.1 Syntax

Consider a countable set Ap of propositions. We define the language of logic $K^{\#}$ as the set of formulas generated by the following BNF:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \xi \geq 0$$
$$\xi ::= c \mid 1_\varphi \mid \#\varphi \mid \xi + \xi \mid c \times \xi$$

The construction $\xi \geq 0$ means that the value of the linear expression ξ is greater than 0. The value 1_φ equals 1 if φ holds, and equals 0 otherwise. The construction $\#\varphi$ is the number of successors satisfying φ.

5.2.2 Semantics

As in modal logic, a formula φ is evaluated in a pointed graph (G, u) (also known as pointed Kripke model). We define the truth conditions $(G, u) \models \varphi$ (φ is true in u) by

$$\begin{array}{lll} (G,u) \models p & \text{if} & \ell(u)(p) = 1, \\ (G,u) \models \neg\varphi & \text{if} & \text{it is not the case that } (G,u) \models \varphi, \\ (G,u) \models \varphi \wedge \psi & \text{if} & (G,u) \models \varphi \text{ and } (G,u) \models \psi, \\ (G,u) \models \xi \geq 0 & \text{if} & [[\xi]]_{G,u} \geq 0, \end{array}$$

and the semantics $[[\xi]]_{G,u}$ (the value of ξ in u) of an expression ξ by mutual induction on φ and ξ as follows.

$$\begin{array}{ll} [[c]]_{G,u} & = c, \\ [[\xi_1 + \xi_2]]_{G,u} & = [[\xi_1]]_{G,u} + [[\xi_2]]_{G,u}, \\ [[c \times \xi]]_{G,u} & = c \times [[\xi]]_{G,u}, \\ [[1_\varphi]]_{G,u} & = \begin{cases} 1 & \text{if } (G,u) \models \varphi \\ 0 & \text{else,} \end{cases} \\ [[\#\varphi]]_{G,u} & = \mathit{card}(\{v \in V \mid (u,v) \in E \text{ and } (G,v) \models \varphi\}). \end{array}$$

We illustrate it in the next example.

Example 83. *Consider the pointed graph G, u shown in Figure 5.1. We have $G, u \models p \wedge (\#\neg p \geq 2\#q) \wedge \#(\#p \geq 1) \leq 1$. Indeed, p holds in u, u has (at least) two successors in which $\neg p$ holds. Moreover, there is (at most) one successor which has at least one p-successor.*

Proposition 84. GML *is a syntactic fragment of $K^{\#}$.*

Proof. $\Diamond^{\geq k}\varphi$ is a shorthand for $\#\varphi \geq k$. □

5.3 $K^{\#}$ and truncReLU-GNNs

In this section, we consider GNNs where the activation function α is truncReLU. We also suppose that the features are in $\{0, 1\}$ (in [Nunn and Schwarzentruber, 2023] and [Nunn et al., 2024], graphs are tagged with propositions while in [Benedikt et al., 2024], vertices have colors).

5.3.1 From $K^{\#}$ to truncReLU-GNNs

The following proposition explains that any $\mathrm{K}^{\#}$-formula can be represented by a AC-GNN. Instead of giving a AC-GNN, we give a AC-GNN-expression for it.

Proposition 85. *For all $K^{\#}$-formulas φ, there is a AC-GNN-expression $tr(\varphi)$, computable in poly-time in $|\varphi|$ such that*

$$[\![\varphi]\!] = [\![tr(\varphi) \geq 1]\!].$$

Proof. We define two translations functions tr for translating formulas and τ for translating $\mathrm{K}^{\#}$-expressions into AC-GNN-expressions:

$$\begin{aligned}
tr(\mathrm{K}^{\#}\text{-formula}) &= \text{GNN-expression} \\
tr(x_i = 1) &= \mathsf{truncReLU}(x_i) \text{ provided } x_i \text{ takes its value in } \{0,1\} \\
tr(\neg\varphi) &= \mathsf{truncReLU}(1 - tr(\varphi)) \\
tr(\varphi \wedge \psi) &= \mathsf{truncReLU}(tr(\varphi) + tr(\psi) - 1) \\
tr(\vartheta \geq 1) &= \mathsf{truncReLU}(\tau(\vartheta)) \\
\\
\tau(\mathrm{K}^{\#}\text{-expression}) &= \text{GNN-expression} \\
\tau(\#\varphi) &= agg(tr(\varphi)) \\
\tau(\vartheta + \vartheta') &= \tau(\vartheta) + \tau(\vartheta') \\
\tau(1_\varphi) &= tr(\varphi) \\
\tau(c) &= c \\
\tau(c \times \vartheta) &= c \times \tau(\vartheta)
\end{aligned}$$

Note that as constants c are integers, the weights in the produced GNN $tr(\varphi)$ are integers. So $\tau(\vartheta)$ is always interpreted as an integer. Hence the following property by definition of tr.

Property 86. *For all $K^{\#}$-formulas φ, $[\![tr(\varphi)]\!]_{G,u} \in \{0, 1\}$.*

We finish the proof by proving by mutual induction on φ and on ϑ that:

1. $G, u \models \varphi$ iff $[\![tr(\varphi)]\!]_{G,u} = 1$;
2. $[\![\vartheta]\!]_{G,u} = [\![\tau(\vartheta)]\!]_{G,u}$.

- Base case. $G, u \models x_i = 1$ iff $[\![x_i]\!]_{G,u} = 1$;
- Inductive case for negation:

$$\begin{aligned}
G, u \models \neg\varphi &\text{ iff } G, u \not\models \varphi \\
&\text{iff } [\![tr(\varphi)]\!]_{G,u} \neq 1 \\
&\text{iff } [\![tr(\varphi)]\!]_{G,u} = 0 \\
&\text{iff } [\![\mathsf{truncReLU}(tr(\varphi))]\!]_{G,u} = 0 \\
&\text{iff } [\![tr(\neg\varphi)]\!]_{G,u} = 1.
\end{aligned}$$

- Inductive case for inequality:

$$\begin{aligned} G, u \models \vartheta \geq 1 \text{ iff } & [\![\vartheta]\!]_{G,u} \geq 1 \\ \text{iff } & [\![\tau(\vartheta)]\!]_{G,u} \geq 1 \\ \text{iff } & [\![\mathsf{truncReLU}(\tau(\vartheta))]\!]_{G,u} = 1 \\ \text{iff } & [\![tr(\vartheta \geq 1)]\!]_{G,u} = 1 \end{aligned}$$

- Inductive case for the counting modality:

$$\begin{aligned} [\![\#\varphi]\!]_{G,u} &= \mathit{card}(\{v \in V \mid (u,v) \in E \text{ and } (G,v) \models \varphi\}) \\ &= \mathit{card}(\{v \in V \mid (u,v) \in E \text{ and } [\![tr(\varphi)]\!]_{G,v} = 1\}) \\ &= \sum_{v|uEv} [\![tr(\varphi)]\!]_{G,v} \\ &= [\![agg(tr(\varphi))]\!]_{G,u} \end{aligned}$$

□

5.3.2 From truncReLU-GNNs to $K^{\#}$

The following translation function tr' takes a GNN with truncReLU as an activation function and rational weights, described as an expression ϑ, and gives an equivalent $K^{\#}$-expression.

Proposition 87. *For all nice GNN-expressions ϑ, there is a $K^{\#}$-expression such that $[\![\vartheta \geq 1]\!] = [\![tr'(\vartheta) \geq 1]\!]$. Furthermore, tr' is computable in poly-time in ϑ* ***and the common denominator of weights (only good if it is represented in unary).***

Proof. We first prove the proposition when weights are integers because it is easier. Then we focus on weights that are rational.

1. Proof when weights are integers. As done in [Nunn and Schwarzentruber, 2023] and [Nunn et al., 2024].

tr'(expression for a GNN)= $\mathrm{K}^{\#}$-expression

$$\begin{aligned} tr'(x_i) &= x_i \\ tr'(c) &= c \\ tr'(c\vartheta) &= c \times tr'(\vartheta) \\ tr'(\vartheta + \vartheta') &= tr'(\vartheta) + tr'(\vartheta') \\ tr'(\mathsf{truncReLU}(\vartheta)) &= 1_{tr'(\vartheta) \geq 1} \\ tr'(agg(\vartheta)) &= \#(tr'(\vartheta) \geq 1) \text{ provided } \vartheta \text{ is of the form } \mathsf{truncReLU}(.) \end{aligned}$$

We prove by induction on ϑ that $[\![\vartheta]\!]_{G,u} = [\![tr'(\vartheta)]\!]_{G,u}$.

2. Rational weights. So far we handled GNNs with integer weights. In order to handle weights that are rationals, consider M a common denominator. Let us analyse the computation of a GNN.

- $N^{(0)}(G,u) = \ell_0(u)$;
- $N^{(t)}(G,u) = \vec{\alpha}(A_t \times N^{(t-1)}(G,u) + B_t \times \sum\{\!\{N^{(t-1)}(G,v) \mid v \in E(u)\}\!\} + b_t)$ for all $u \in V$.

By induction on t, we prove that that $N^{(t)}(G,u) \in \{0, \frac{1}{M^t}, \frac{2}{M^t}, \ldots, 1\}$. For instance for $M = 2$ and $t = 3$ we could have:

$$\frac{1}{2}\mathsf{truncReLU}(\frac{1}{2} \times \mathsf{truncReLU}(\frac{1}{2} \times 1)) = \frac{1}{2^3} = \frac{1}{8}.$$

So at round 3, we have to multiply by 8 to get an integer value. More generally, the idea is then to multiply all the numbers at round t by M^t and to simulate the activation function truncReLU by:

$$x \mapsto \max(0, \min(M^t, x)).$$

The translation is then:

tr'_t(expression for a GNN)= $\mathrm{K}^{\#}$-expression

$$\begin{aligned}
tr'_t(\ell_i) &= M^t \times \ell_i \\
tr'_t(c) &= M^t \times c \\
tr'_t(c\vartheta) &= M \times c \times tr'_{t-1}(\vartheta) \\
tr'_t(\vartheta + \vartheta') &= tr'_t(\vartheta) + tr'_t(\vartheta') \\
tr'_t(\mathsf{truncReLU}(\vartheta)) &= 1_{tr'_t(\vartheta)=1} + 2 \times 1_{tr'_t(\vartheta)=2} + \cdots + M^t \times 1_{tr'_t(\vartheta)\geq M^t} \\
tr'_t(agg(\vartheta)) &= \#(tr'_t(\vartheta) = 1) + 2 \times \#(tr'_t(\vartheta) = 2) + \ldots M^t \times \#(tr'_t(\vartheta) = M^t) \\
&\qquad \text{provided } \vartheta \text{ is of the form } \mathsf{truncReLU}(.)
\end{aligned}$$

Fact 88. $[\![tr'_t(\vartheta)]\!]_{G,u} = M^t \times [\![\vartheta]\!]_{G,u}$.

Fact 89. *For all AC-GNN-*L*-expressions* ϑ *with* L *layers,* $[\![\vartheta \geq 1]\!] = [\![tr'_L(\vartheta) \geq M^L]\!]$.

To sum up, we define $tr'(N)$ to be $tr'_L(N)$ where L is the number of layers in the GNN N. □

Remark 90. *If* M *is written in unary, there is no exponential blow-up. But written in binary there is one. The idea in the algorithm we will see at the end that it is sufficient to guess the value of* $tr'(\vartheta)$ *between 0 and* M *for the* truncReLU *case. But there is still a blow-up problem for agg. Is it harmful? We will see later, in Section 8.6.*

In [Benedikt et al., 2024], the authors generalizes the methodology to capture any activation functions that is piece-wise linear and eventually constant.

5.4 Reduction to the satisfiability of $K^{\#}$

We explain how to solve problem 4 ($[\![\varphi]\!] \subseteq [\![N]\!]$). First we suppose that φ is already in $K^{\#}$. Then we use Proposition 87 to get $\tau'(N)$. We then check that $\varphi \to \tau'(N)$ is $\mathrm{K}^{\#}$-valid, i.e. that $\neg(\varphi \to \tau'(N))$ is not $\mathrm{K}^{\#}$-satisfiable.

5.5 Going further

5.5.1 ReLU

Handling ReLU is more involved, and we do not have a clear correspondence with modal logic yet [Benedikt et al., 2024].

In [Barceló et al., 2026], they consider a layer that can use truncReLU and/or the identity function (i.e. purely linear layers) in the GNN. The expressivity then goes beyond $\mathrm{K}^{\#}$. In particular, we handle properties like 'the number of red grandchildren is greater than the number of blue grandchildren'... $\#\#red \geq \#\#blue$ which cannot be expressed in $\mathrm{K}^{\#}$.

Proposition 91. *$\#\#red \geq \#\#blue$ can be expressed by a ReLU-AC-GNN.*

Proof. Red and blue are $0-1$ properties. With aggregation we can count. With ReLU and an other aggregation we transfer the counting a level further. Here is a GNN (expressed as a AC-GNN-expression) that classify pointed graphs satisfying $\#\#red \geq \#\#blue$:

$$\mathsf{ReLU}(agg(\mathsf{ReLU}(agg(red)))) - \mathsf{ReLU}(agg(\mathsf{ReLU}(agg(blue)))) \geq 0.$$

□

In [Haase and Zetzsche, 2019], they generalize existential Presburger arithmetics with Kleene star as follows. Kleene star is the mechanism to perform sums of vectors, and thus to express the sum aggregation function. The Kleene star of M contains any finite sum of vectors from M. The formal definition is as follows.

Definition 92. *The* Kleene star *of a set $M \subseteq \mathbb{Z}^d$ of vectors is defined by:*

$$M^* := \bigcup_{k \geq 0} \left\{ \sum_{i=1}^{k} v_i \mid v_i \in M \right\}.$$

Example 93. *The Kleene star of $\{(1,1),(2,0)\}$ is: $\{k(1,1) + n(2,0) \mid k, m \in \mathbb{N}\}$.*

We extend now Presburger arithmetic with Kleene star. The obtained logic is *Existential Presburger arithmetic with star* $\exists\mathsf{PA}^*$. The syntax is:

$$\varphi, \psi, ... \ ::= \ \vec{a} \cdot \vec{z} \geq c \ \mid \ \varphi \wedge \psi \ \mid \ \varphi \vee \psi \ \mid \ \exists y \varphi \ \mid \ \varphi^*$$

The semantics $[\![\varphi]\!]$ for φ is the set of vectors of values in $\mathbb{Z}$ such that φ is true when we replace the free variables in φ by these values. By induction on φ, we define $[\![\varphi]\!]$ (w.l.o.g. we suppose that φ and ψ contains the same free variables for $\varphi \vee \psi$ and $\varphi \wedge \psi$):

$$\begin{aligned} [\![\vec{a} \cdot \vec{z} \geq c]\!] &= \{\vec{x} \in \mathbb{Z}^n \mid \vec{a} \cdot \vec{x} \geq c\} \\ [\![\varphi \vee \psi]\!] &= [\![\varphi]\!] \cup [\![\psi]\!] \\ [\![\varphi \wedge \psi]\!] &= [\![\varphi]\!] \cap [\![\psi]\!] \\ [\![\varphi^*]\!] &= [\![\varphi]\!]^* \end{aligned}$$

Example 94. $[\![2x \geq 10]\!]$ *is* $\{5, 6, 7, 8, \dots\}$.

Example 95. $[\![\exists z, z = 2\ell - 4\ell' \wedge [(z \geq 0) \wedge (x = 0) \vee [(z \geq 0) \wedge (x = z)]]\!]$ *is* $\{0, 2, 4, 6, \dots\}$.

Theorem 96. *(Th. III.1 in [Haase and Zetzsche, 2019]) The satisfiability problem of* $\exists\mathsf{PA}^*$ *is* NEXPTIME*-complete, and* NP*-complete if the number of nested star is bounded.*

Corollary 97. *(Th. 6.20 in [Benedikt et al., 2024]) The satisfiability problem of some* ReLU*-AC-GNNs is* NEXPTIME*-complete, and* NP*-complete if the number of layers is bounded.*

Proof. Lower bound Lower bound (Th. 6.28 in [Benedikt et al., 2024]) is proven by reducing the $\exists\mathsf{PA}^*$-satisfiability to the satisfiability of a GNN with ReLU. We reduce to the satisfiability of $\exists\mathsf{PA}^*$.

Upper bound We only reproduce the proof of the upper bound here (Th. 6.26 in [Benedikt et al., 2024]). The Kleene-star is used to perform the aggregation over an unbounded number of successors.

Consider a GNN N (w.l.o.g we suppose that the weights are integers; if not, multiply by a common denominator). We define formula $\varphi_t(\vec{x}_0, \ldots, \vec{x}_t)$ that says that $x_0, x_1, \ldots, x_t$ are the values of $N^{(0)}(G,u), \ldots, N^{(t)}(G,u)$ for some pointed graph G, u.

$$\varphi_0(\vec{x}_0) = \bigwedge_{i=1..d} (x_{0,i} = 0) \vee (x_{0,i} = 1) \tag{5.1}$$

$$\varphi_t(\vec{x}_0, \ldots, \vec{x}_t) = \varphi_0(\vec{x}_0) \wedge \tag{5.2}$$

$$\exists \overrightarrow{agg}\vec{v}_0, \ldots \exists \overrightarrow{agg}\vec{v}_{t-1} \tag{5.3}$$

$$(\varphi_{t-1}(\overrightarrow{agg}\vec{v}_0, \ldots, \overrightarrow{agg}\vec{v}_{t-1}))^* \tag{5.4}$$

$$\wedge\, \exists \vec{z}_1, \ldots \exists \vec{z}_t \tag{5.5}$$

$$\bigwedge_{\tau=1..t} (\vec{z}_\tau = A_\tau \vec{x}_{\tau-1} + B_\tau \overrightarrow{agg}\vec{v}_{\tau-1} + b_\tau) \tag{5.6}$$

$$\wedge \bigwedge_i (z_{\tau,i} \leq 0) \wedge (x_{\tau,i} = 0) \vee (z_{\tau,i} \geq 0) \wedge (x_{\tau,i} = z_{\tau,i}) \tag{5.7}$$

Equation (6.1) says that the initial value of each feature should be 0 or 1. For timestep $t = 0$, the relevant part of some pointed graph G, u is just the feature vectors at u.

In (6.3), we existentially quantify for vectors $\overrightarrow{agg}\vec{v}_0, \ldots, \overrightarrow{agg}\vec{v}_{t-1}$, that are the results of the aggregation at step $0, \ldots, t-1$.

The part (6.4) is more subtle and explain the results of the aggregation (sum). Note that we grouped all the values so that the structure – the successors of G, u – is guessed at once in the Kleene star (if we would have several Kleene stars, we would have several independent sets of "successors"). The vector $\vec{z}_\tau$ is the one obtained just after the combination (see 6.6), and before the activation function. In (6.7), we wrote the fact that $x_{\tau,i} = ReLU(z_{\tau,i})$.

Our final formula is:

$$tr(N) := \exists \vec{x}_0 \ldots \exists \vec{x}_L \varphi_L(x_0, \ldots, x_L) \wedge (w^t \times x_L + b \geq 0).$$

We have that there is a pointed graph G, u such that $N(G,u) = true$ iff $tr(N)$ is $(PA)^*$-satisfiable. □

Our AC-GNN-L and $\exists\mathsf{PA}^*$ are very similar, but they also differ. The semantics of a expression is a single real number, whereas the semantics of a $\exists\mathsf{PA}^*$ formula is a list of values (one for each free variable). Having a single real number requires to evaluate wrt to a pointed graph G, u: we need to have the structure somewhere, to be sure that all $agg(....)$,

the evaluation is according the same structure. In the semantics $\exists\mathsf{PA}^*$ the structure is implicit. If we have several formulas $(..)^*$, then the successors are different. That is why in [Benedikt et al., 2024] they pack all the relevant values in the same list, to have a single formula $(...)^*$, i.e. to have the a single set of successors for a given vertex.

5.5.2 Panorama

We reproduce a (simplified version of the) table from [Benedikt et al., 2024] that sums up the situation for the satisfiability problem of a GNN.

	searching for directed graphs	searching for undirected graphs
truncReLU	PSPACE-complete	PSPACE-complete
ReLU	NEXPTIME-complete	undecidable

5.5.3 Quantized GNNs

In real implementations, GNNs are quantized. For instance, numbers are 32-bit floats. In [Sälzer et al., 2025], the authors prove that the satisfiability problem of AC-GNN-L when numbers are quantized is in PSPACE-complete. The bitwidth n is the number of bits used to represent the numbers.

Theorem 98. *Verifying quantized GNNs where the bitwidth n is given in unary is PSPACE-complete.*

Open questions

- Nice logic for truncReLU without the exponential blow-up due to rational numbers. In [Benedikt et al., 2024], they seem to treat the problem algorithmically but not with a poly-time translation in a logic
- Parametrized complexity of $\exists\mathsf{PA}^*$ wrt to the nested number of stars?
- Design a "neat" modal logic that is equivalent to GNN with ReLU?
- Parametrized complexity of verifying quantized GNNs wrt to the bitwidth
- Lower bounds for other activation functions than truncReLU for quantized GNNs
- Proof systems for the introduced logics

Exercises

Exercise 15. *Write formally the translation from* GML *into* $K^\#$ *preserving the semantics.*

Exercise 16. *Prove that* $K^\#$ *is more expressive than* FO.

Hint: see Appendix in [Nunn et al., 2024]

Exercise 17. *Prove that we can reduce in poly-time the satisfiability problem of* $K^\#$ *to the satisfiability problem of a* $K^\#$*-formula without any occurrence of* 1_ψ.

Hint: see [Nunn et al., 2024]

Exercise 18. *Fix* $L \in \mathbb{N}$. *What is the complexity of the following problem:*

- *input: a* $K^\#$*-formula* φ *of modal depth at most* L*;*
- *output: yes if* φ *is satisfiable.*

Exercise 19. *Prove that* $\#\#red \geq \#\#blue$ *cannot be expressed in* $K^\#$.

Chapter 6

Satisfiability problem of modal logic

Before tackling the satisfiability problem for graded modal logic (and the richer logic $K^{\#}$ introduced in Chapter 5), it is good to concentrate on a simple setting. In this chapter, we tackle the satisfiability problem for basic modal logic K:

- input: a modal formula φ;
- output: yes if φ is satisfiable; no otherwise.

We will explain the tableau method, an algorithm for deciding satisfiability problem.

6.1 Negative normal form

In the tableau method we will propose, we need disjunction, conjunction, box and diamonds are *explicit*, e.g. no disjunction is hidden like in $\neg(\varphi \wedge \psi)$! That is why we introduce the notion of formula in *negative normal form* where negations only appear in front of atomic propositions.

Definition 99 (negative normal form)**.** *A formula in* negative normal form *belongs to the language defined by the rule*

$$\varphi \ ::= \ p \ \mid \ \neg p \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \Diamond\varphi \ \mid \ \Box\varphi$$

where p ranges over the set of atomic propositions.

We suppose that the formula φ (and all formulas) are in *negative normal form.* If φ is not in negative normal form, apply these rewriting rules that pushes negations in front of atomic propositions:

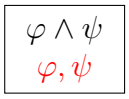
$\neg\Box\varphi$	becomes	$\Diamond\neg\varphi$
$\neg\Diamond\varphi$	becomes	$\Box\neg\varphi$
$\neg(\varphi \wedge \psi)$	becomes	$(\neg\varphi \vee \neg\psi)$
$\neg(\varphi \vee \psi)$	becomes	$(\neg\varphi \wedge \neg\psi)$

6.2 Overview

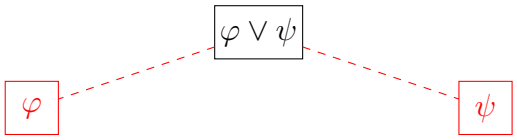
In a nutshell, the tableau method is a proof system that constructs a Kripke structure. Each time a formula is written, it means that the formula *should* hold at a given world. It can be seen as a procedure that rewrites a labelled graph. The tableau method starts with an initial graph made up of one node containing φ:

6.3 Tableau rules

Tableau rules there are rewriting rules that make explicit the meaning of 'a formula *should* hold'. Let us start with the rule for $\wedge$ (in red, we write what is added).



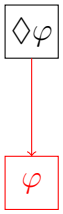
The following rule for the $\vee$ connective is non-deterministic.



The clash rule for contradiction gives a clash.

clash

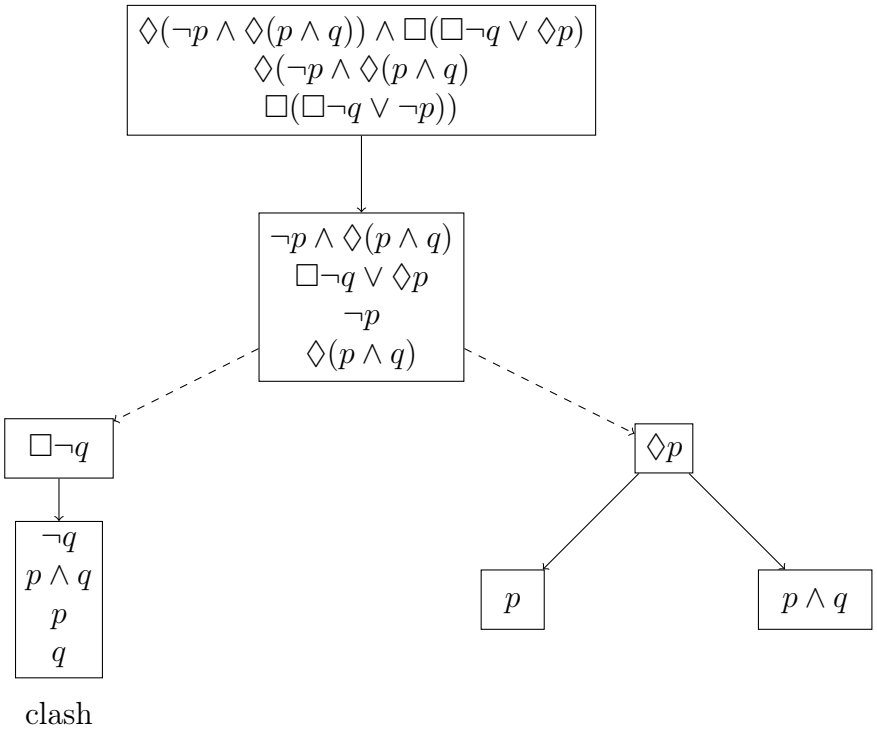
The rule for $\Diamond$ adds a successor containing φ if there is no successor.



The rule for $\Box$ adds φ in all successors.

6.4 Example

We draw dashed arrow for non-determinism and plain arrow for the relation in the model that is created.



6.5 Tableau system as a labelled proof system

A tableau method works (and can be formalized) as a rewriting term system. At each time of the algorithm we maintain a set of terms of the following form:

- $(\sigma\ \varphi)$, where σ is a abstract symbol and φ is a formula. The term $(\sigma\ \varphi)$ means that 'φ should be true in the world denoted by σ'.
- $(R\ \sigma\ \sigma')$ where σ and σ' are two symbols. The term $(R\ \sigma\ \sigma')$ means that 'the world denoted by σ is linked by relation R to the world denoted by σ'.

The tableau method starts with $(\sigma\ \varphi)$ where σ is a fresh symbol and φ is the formula we want to test. We then apply some rules.

6.5.1 Boolean tableau rules

Let us start by defining the Boolean tableau rules for Boolean connectives:

$$\frac{(\sigma\ \varphi_1 \wedge \varphi_2)}{(\sigma\ \varphi_1)(\sigma\ \varphi_2)}\ \text{(Rule } \wedge\text{)} \qquad \frac{(\sigma\ \varphi_1 \vee \varphi_2)}{(\sigma\ \varphi_1) \mid (\sigma\ \varphi_2)}\ \text{(Rule } \vee\text{)}$$

$$\frac{(\sigma\ p)(\sigma\ \neg p)}{\text{rejecting execution}}\ \text{(Clash rule)}$$

Another presentation is to handle a set Γ of formulas. The set Γ corresponds to a given σ: it contains the formulas φ such that $(\sigma\ \varphi)$ is produced. The rules are then:

- If $\varphi_1 \wedge \varphi_2$ is in Γ, then add φ_1 and φ_2 to Γ;
- If $\varphi_1 \vee \varphi_2$ is in Γ, then non-deterministically choose to either add φ_1 in Γ, or add φ_2 in Γ;
- If p and $\neg p$ are Γ, then the execution is rejecting.

6.5.2 Tableau rules for modal operators

Now we define the rules for modal operators:

$$\frac{(\sigma\ \Diamond\varphi)}{(R\ \sigma\ \sigma_{\text{new}})(\sigma_{\text{new}}\ \varphi)}\ (\text{Rule}\ \Diamond) \qquad \frac{(\sigma\ \Box\varphi)(R\ \sigma\ \sigma')}{(\sigma'\ \varphi)}\ (\text{Rule}\ \Box)$$

where σ_{new} is new fresh symbol.

6.6 Soundness and completeness

Theorem 100 (soundness)**.** *If there is an execution of the tableau method that is not clashing, then the initial formula is satisfiable.*

Theorem 101 (completeness)**.** *If the initial formula is satisfiable then there is an execution of the tableau method that is not clashing.*

We leave the proofs of these theorems to the next chapter.

Bibliographical notes

In [Blackburn et al., 2001] (p. 383), the definition of Hintikka set [Blackburn p. 357, Def 6.24] is given. Their algorithm is cleaner in a sense, but not as easy to understand. The tableau method presented here can be extended for graded modal logic [Tobies, 1999].

Concerning implementation issues and optimisation, the reader may have a look to [Horrocks et al., 2007].

Exercices

1. Apply the tableau method to the modal formula of your choice.
2. Explain informally why any satisfiable modal formula is satisfiable in a tree. Can you bound its arity? its depth?
3. How would you adapt the tableau method to know whether a given formula is true in a reflexive model (uRu for all worlds u)?

Chapter 7

Satisfiability problem of modal logic is PSPACE-complete

Savitch theorem says that PSPACE = NPSPACE. It means that proving a PSPACE upper bound can be proven by given a non-deterministic algorithm that requires a polynomial amount of space. We now turn the tableau method of Chapter 6 we just saw into a non-deterministic algorithm.

7.1 What do we mean by a non-deterministic algorithm?

A *non-deterministic algorithm* is like a deterministic algorithm (i.e. implementable on a real computer) but it has a one-player game flavor. The player has a choose operation.

An input is accepted by the algorithm if the player has a *winning strategy*. Otherwise, the input is rejected.

7.2 Algorithm

The design of our algorithm is as follows.

- Boolean rules are applied in a given node are performed in the same call (see Section 6.5.1);
- Rules for modal operators are simulated by recursive calls. The number of recursive calls is thus bounded by the modal depth.

procedure satK(Γ)
apply non-deterministically Boolean tableau rules to Γ until no more rules applicable:

- if $\varphi \wedge \psi \in \Gamma$, remove $\varphi \wedge \psi$ from Γ and add φ and ψ to Γ
- if $\varphi \vee \psi \in \Gamma$, remove $\varphi \vee \psi$ from Γ and choose either to add φ to Γ or add ψ to Γ
- if φ and $\neg\varphi$, **reject**

for all formulas of the form $\Diamond\psi$ in Γ
satK($\psi \cup \{\chi \mid \Box\chi \in \Gamma\}$)

7.3 Soundness and completeness

We denote the modal depth of φ by $md(\varphi)$. We define $md(\Gamma) = max_{\psi\in\Gamma} md(\psi)$.

Theorem 102 (completeness)**.** *If* Γ *is satisfiable, then there exists an non-rejecting execution of* $sat(\Gamma)$*.*

Proof. By induction on $md(\Gamma)$. Let $P(k)$ is

'For all Γ such that $md(\Gamma) \leq k$, if Γ is satisfiable, then there exists a non-rejecting execution of $sat(\Gamma)$.'

$\boxed{md(\Gamma) = 0}$ There exists a model $\mathcal{M} = (W, R, V)$ and a world w such that $\mathcal{M}, w \models \psi$ for all Γ. We prove the following invariant during one possible execution of the algorithm:

$$\text{for all } \psi \in \Gamma,\ \mathcal{M}, w \models \psi.$$

Rule and: if $\psi_1 \wedge \psi_2 \in \Gamma$, then $\mathcal{M}, w \models \psi_1 \wedge \psi_2$. The rule and adds $\psi_1, \psi_2 \in \Gamma$. By the definition of the truth condition, $\mathcal{M}, w \models \psi_1$ and $\mathcal{M}, w \models \psi_2$.

Rule or: if $\psi_1 \vee \psi_2 \in \Gamma$, then $\mathcal{M}, w \models \psi_1 \vee \psi_2$. Either $\mathcal{M}, w \models \psi_1$ or $\mathcal{M}, w \models \psi_2$. Suppose that we are in the case where $\mathcal{M}, w \models \psi_2$. It is then sufficient to consider the execution where ψ_2 is added to Γ and the invariant remains true.

As $\mathcal{M}, w \models p$ and $\mathcal{M}, w \models \neg p$ is impossible, the execution is non-rejecting.

$\boxed{\text{recursive case}}$

Suppose $P(k)$ and let us prove $P(k+1)$. Let Γ such that $md(\Gamma) = k+1$. There exists a model $\mathcal{M} = (W, R, V)$ and a world w such that $\mathcal{M}, w \models \psi$ for all Γ. The beginning of the proof is the same than for the basic case: we make the non-deterministic choices according to the truth of formulas in w.

Now, for all formulas of the form $\Diamond\psi$ in Γ, as $\mathcal{M}, w \models \Diamond\psi$ there exists $u \in R(w)$ such that $\mathcal{M}, u \models \Diamond\psi$. More: we have $\mathcal{M}, u \models \chi$ for all $\Box\chi \in \Gamma$.

Thus, $\psi \cup \{\chi \mid \Box\chi \in \Gamma\}$ is satisfiable. By $P(k)$, satK$(\psi \cup \{\chi \mid \Box\chi \in \Gamma\})$ has an non-rejecting execution. We can thus construct an non-rejecting execution of satK(Γ). □

Theorem 103 (soundness)**.** *If* $sat(\Gamma)$ *has an non-rejecting execution, then* Γ *is satisfiable in a tree of depth* $md(\Gamma)$ *and of arity the number of* $\Diamond$ *that appears in* Γ*.*

Proof. $P(k)$ is defined as:

If $sat(\Gamma)$ has an non-rejecting execution, then Γ is satisfiable in a tree of depth $md(\Gamma)$ and of arity the number of $\Diamond$ that appears in Γ.

$\boxed{\text{basic case}}$

There is no modal operator. We define a model $\mathcal{M}$ made up of a unique world w and the valuation $V(w) = AP \cap \Gamma$ when the saturation has been made.

We prove that by induction on $\varphi \in \Gamma$ that $\mathcal{M}, w \models \varphi$.

Propositions: ok

Negations of proposition: ok

Or: If $\varphi \vee \psi \in \Gamma$, we have either φ or $\psi \in \Gamma$ because all the Boolean rules has been applied. Suppose it is $\psi \in \Gamma$. We have $\mathcal{M}, w \models \psi$. Thus, $\mathcal{M}, w \models \varphi \vee \psi$.

And: idem.

$\boxed{\text{recursive case}}$ Suppose $P(k)$. Let us prove $P(k+1)$.

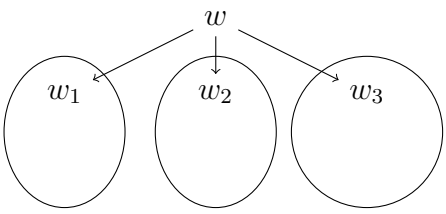


Figure 7.1: Model constructed by gluing models obtained from the subcall when diamond formulas are $\Diamond\psi_1, \Diamond\psi_2$ and $\Diamond\psi_3$.

Let us take Γ of model depth $k{+}1$ such that $sat(\Gamma)$ succeeds. Then satK$(\psi \cup \{\chi \mid \Box\chi \in \Gamma\})$ succeeds for all $\Diamond\psi \in \Gamma$ after Boolean saturation.

As shown in Figure 7.1, we construct $\mathcal{M}$ by gluing models obtained from the subcall. By induction, for all $\Diamond\psi \in \Gamma$, we can find a tree $\mathcal{M}_\psi$ of depth at most k and arity at most the number of $\Diamond$ in $\psi \cup \{\chi \mid \Box\chi \in \Gamma\}$... well Γ.

We then create a root w as in the basic case where $V(w) = AP \cap \Gamma$. We prove that by induction on $\varphi \in \Gamma$ that $\mathcal{M}, w \models \varphi$.

□

7.4 PSPACE upper bound

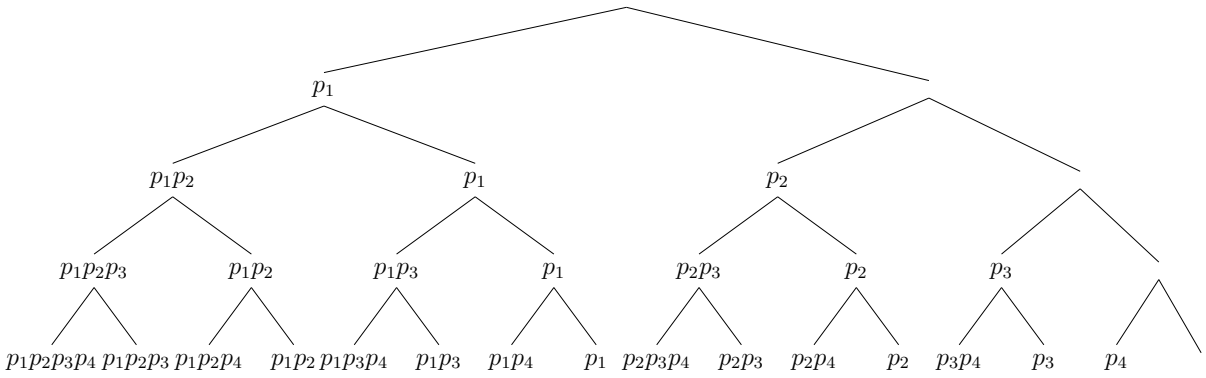
Theorem 104. *The satisfiability problem for K is in PSPACE.*

Proof. The algorithm we saw is sound and complete. It runs in polynomial space and is non-deterministic. So the satisfiability problem for K is in NPSPACE. By Savitch's theorem, NPSPACE = PSPACE. □

7.5 PSPACE lower bound

Theorem 105. *The satisfiability problem for K is in PSPACE-hard.*

Proof. By reduction from TQBF. Let us take a quantified binary formula $\exists p_1 \forall p_2 \ldots \exists p_{2n-1} \forall p_{2n} \chi$ where χ is propositional. The game behind TQBF can be represented by the binary tree in which p_i is chosen at the i-th level. Player $\exists$ plays at the root level (level 0), at level 2, at level 4, etc. while player $\forall$ responds at level 1, at level 3, etc. The leaves correspond to all propositional valuations. Once we reach a leaf, the game ends and player $\exists$ wins iff the corresponding valuation satisfies χ.



What is nice is that modal logic can express that a Kripke model contains the above tree:

$$TREE := \bigwedge_{i=1}^{2n} \Box^{i-1}[\Diamond p_i \wedge \Diamond \neg p_i \wedge \bigwedge_{j<i}(p_j \leftrightarrow \Box p_j) \wedge (\neg p_j \leftrightarrow \Box \neg p_j)].$$

Formula $TREE$ explains explains how each i-th level works:

- With $\Box^{i-1} = \Box \ldots \Box$, we reach the $i-1$-th level;
- $\Diamond p_i \wedge^{\Diamond} \neg p_i$ enforces the existence of a p_i-child and a $\neg p_i$-child;
- $\bigwedge_{j<i}(p_j \leftrightarrow \Box p_j) \wedge (\neg p_j \leftrightarrow \Box \neg p_j)$ says the values of $p_1, \ldots, p_{i-1}$ are copied to the successors.

On the input $\varphi := \exists p_1 \forall p_2 \ldots \exists p_{2n-1} \forall p_{2n} \chi$, the reduction computes in polynomial time the modal formula $tr(\varphi) := TREE \wedge \Diamond\Box ... \Diamond\Box\chi$. The former φ is QBF-true iff the latter $tr(\varphi)$ is K-satisfiable. Furthermore, tr is computable in poly-time. $\Box$

Exercices

1. Write a deterministic algorithm for deciding the satisfiability for K.

   Hint: use backtracking.

2. S5 is the modal logic interpreted in Kripke models where the relation is universal.

   (a) Prove that if φ is S5-satisfiable, then it is true in a model in which the number of worlds is bounded by the number of modal operators in φ.

   (b) Deduce that the satisfiability for S5 is in NP.

   (c) Why is the satisfiability for S5 NP-hard?

3. Prove that K is PSPACE-hard even with 0 variables! See [Chagrov and Rybakov, 2002]

4. Adapt the algorithm for the satisfiability for S4, the modal logic interpreted in Kripke models where the relation is reflexive and transitive. (difficult)

Chapter 8

Satisfiability problem of modal logic with counting

In this chapter, we focus on the satisfiability problem of a $K^{\#}$-formula.

8.1 Difficulty to get a PSPACE Tableau Method for $K^{\#}$

As explained in [Nunn and Schwarzentruber, 2023], it is not sufficient, as for K, to prove consistency in successors. We also have to take into account *implicit* counting relations. For instance, we always have:

$$\#p \ + \ \#\neg p \quad = \quad \#q \ + \ \#\neg q.$$

To take these constraints into account, we rely on QFBAPA (Quantifier-free Fragment Boolean Algebra with Presburger Arithmetic) which combines Presburger arithmetic (reasoning about linear inequalities) and Boolean algebra (reasoning about p, $\neg p$, q, $\neg q$, and all Boolean formulas). We will not use the full power of QFBAPA.

8.2 Venn diagrams

In order to avoid to reason about implicit counting relations, we rely on the notion of Venn diagrams. We explain the idea with that example.

Example 106.

$$\begin{cases} \#\psi_1 + 3\#\psi_2 \leq 5 \\ 2\#\psi_1 + 4\#\psi_3 \leq 42 \end{cases}$$

Here the subsets correspond to properties ψ_1, ψ_2, ψ_3. In general we have $\psi_1, \ldots, psi_d$.

A Venn diagram is a picture that contains all the possible intersections called *regions*, see Figure 8.1. Each region is denoted by a *region code* $\rho \in \{0,1\}^d$. For instance, the code of $\neg\psi_1 \wedge \neg\psi_2 \wedge \psi_3$ is 001.

We introduce integer variable $s_{01010110}$ to denote the size of region 01010110. Now the idea is simple: just replace $\#\psi$ to the sum of the sizes of regions in which ψ holds: replace $\#\psi_i$ by $\sum_{\rho \in \mathbb{B} | \rho_i = 1} s_\rho$.

Example 107. *For instance, $\#\psi_2$ is replaced by $s_{010} + s_{011} + s_{110} + s_{111}$.*

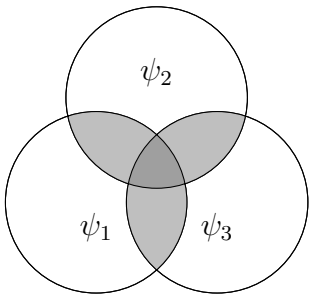


Figure 8.1: A Venn diagram with 3 sets is made up of 8 regions.

8.3 Naïve algorithm

input: Γ a finite set of $K^{\#}$-formulas
output: has a non-rejecting execution iff Γ is $K^{\#}$-satisfiable
procedure sat$K^{\#}(\Gamma)$

for non-#-nested expressions 1_φ **do**
choose either to add φ in Γ and replace 1_φ by 1 in Γ
or to add $\neg\varphi$ in Γ and replace 1_φ by 0 in Γ

apply non-deterministically Boolean tableau rules to Γ until no more rules applicable:

- if $\varphi \wedge \psi \in \Gamma$, remove $\varphi \wedge \psi$ from Γ and add φ and ψ to Γ
- if $\varphi \vee \psi \in \Gamma$, remove $\varphi \vee \psi$ from Γ and choose either to add φ to Γ or add ψ to Γ
- if φ and $\neg\varphi$, clash

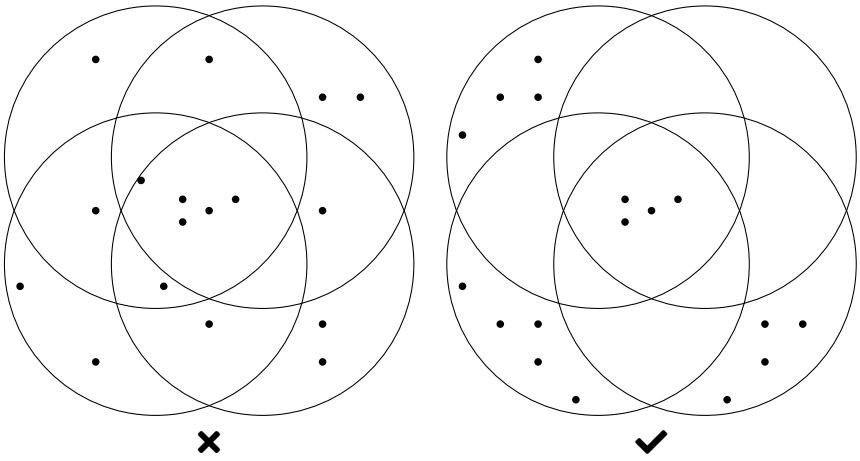
let $\#\psi_1, \ldots \#\psi_d$ be a list of all the non-#-nested constructions of the form $\#\psi$ in Γ
choose a subset $\mathbb{B} \subseteq \{0,1\}^d$
Replace each occurrence of $\#\psi_i$ by $\sum_{\rho \in \mathbb{B} | \rho_i = 1} s_\rho$ in Γ
let $\mathcal{S}$ be the set of inequalities in Γ
check that $\mathcal{S}$ is ILP-satisfiable
for all regions $\rho \in \mathbb{B}$ **do**
sat$K^{\#}(\{\neg\psi_i \mid \rho_i = 0\} \cup \{\psi_i \mid \rho_i = 1\})$

Trust me. This algorithm is correct but runs in non-deterministic exponential time. There is an exponential number of regions in d.

8.4 Polynomial upper bound on the number of non-zero regions

Up to now, we are not finished since there are an exponential number of variables s_ρ. We will show via a Carathéodory bound that if there is a solution, then it is possible to get a

solution in which only a polynomial number of s_ρ is non-zero. Furthermore, the non-empty regions are among the previous non-empty ones.



8.4.1 Carathéodory bound

We will apply a Carathéodory bound for integer cones, see Th. 1 (ii) in [Eisenbrand and Shmonin, 2006], reformulated by the following Lemma 109. We define the definition of a cone.

Definition 108. *Given $X \subseteq \mathbb{Z}^d$, we define the* integer cone *of X by:*

$$\texttt{cone}(X) := \{\lambda_1 x_1 + \cdots + \lambda_t x_t \mid t \geq 0, x_1, \ldots, x_t \in X, \lambda_1, \ldots, \lambda_t \in \mathbb{N}\}\,.$$

In the following Lemma 109, for a d-dimensional vector x, we write $||x||_\infty := \max_{i=1..d} |x_i|$. It stands for the magnitude of x. And then M is the magnitude of a subset X of vectors. The following Lemma 109 is a *Carathéodory bound* saying that any vector in the cone can be obtained as a sum of only few vectors in X.

Lemma 109. *[Eisenbrand and Shmonin, 2006] Let $X \subseteq \mathbb{Z}^d$ be a finite subset. Let $M = \max_{x \in X} ||x||_\infty$.*
For all $b \in \texttt{cone}(X)$ there exists $\tilde{X} \subseteq X$ such that $|\tilde{X}| \leq 2d \log_2(4dM)$ and $b \in \texttt{cone}(\tilde{X})$.

Proof. Suppose that $|X| > 2d \log_2(4dM)$ (otherwise we are done). Consider $b \in \texttt{cone}(X)$. The goal is to prove that we can remove at least one vector in X, getting a strictly smaller $\tilde{X}$ while b is still in the cone of $\tilde{X}$.

Fact 110. $d \log_2(2|X|M + 1) \leq |X|$.

Proof. Our assumption $|X| > 2d\log_2(4dM)$ can be rewritten as $M < \frac{2^{|X|/2d}}{4d}$. Therefore:

$$\begin{aligned}
d\log_2(2|X|M+1) &< d\log_2\left(\frac{|X|}{2d}2^{|X|/(2d)}+1\right) && \text{by what we just said}\\
&\leq d\log_2\left(2^{|X|/(2d)}\left(\frac{|X|}{2d}+1\right)\right)\\
&= \frac{|X|}{2} + d\log_2\left(\frac{|X|}{2d}+1\right) && \text{by playing with the } \log_2\\
&\leq \frac{|X|}{2} + d\cdot\frac{|X|}{2d} && \text{by concavity of } \log_2\\
&= |X|.
\end{aligned}$$

□

Suppose that $b = \sum_{x\in X}\lambda_x x$ with $\lambda_x \in \mathbb{N}^*$ for all $x \in X$ (all the λ_x are strictly positive means that X is apparently fully used, otherwise we are done et get a strictly smaller $\tilde{X}$).

For $\tilde{X} \subseteq X$, we have $\sum_{x\in\tilde{X}} x \in \{-|X|M,\ldots,|X|M\}^d$. So

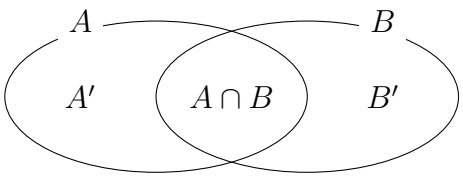
$$\begin{aligned}
card(\left\{\sum_{x\in\tilde{X}} x \mid \tilde{X}\subseteq X\right\}) &\leq card(\{-|X|M,\ldots,|X|M\}^d)\\
&= (2|X|M+1)^d\\
&< 2^{|X|} && \text{by Fact 110}
\end{aligned}$$

Said in words, there are less values for $\sum_{x\in\tilde{X}} x$ than possible subsets for $\tilde{X}$. By the pigeonhole principle, there are two sets $A, B \subseteq X$, $A \neq B$ such that

$$\sum_{x\in A} x = \sum_{x\in B} x.$$

We set:

$$\begin{aligned}
A' &:= A \setminus B\\
B &:= B \setminus A
\end{aligned}$$



Just removing the sum of vectors in $A \cap B$, we get that the sum over A' is equal to the sum over B'. Formally:

$$\sum_{x\in A'} x \;=\; \sum_{x\in A} x - \sum_{x\in A\cap B} x \;=\; \sum_{x\in B} x - \sum_{x\in A\cap B} x \;=\; \sum_{x\in B'} x.$$

We have $A' \neq \emptyset$ or $B' \neq \emptyset$ (otherwise $A = B$!). W.l.o.g. we suppose that $A' \neq \emptyset$. We set $\lambda := \min_{x \in A'} \lambda_x$. Then:

$$
\begin{aligned}
b = \sum_{x \in X} \lambda_x x &= \sum_{x \in X \setminus A'} \lambda_x x + \sum_{x \in A'} \lambda_x x \\
&= \sum_{x \in X \setminus A'} \lambda_x x + \sum_{x \in A'} (\lambda_x - \lambda) x + \lambda \sum_{x \in A'} x \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow = \\
&= \sum_{x \in X \setminus A'} \lambda_x x + \sum_{x \in A'} (\lambda_x - \lambda) x + \lambda \sum_{x \in B'} x \\
&= \sum_{x \in A'} (\lambda_x - \lambda) x + \sum_{x \in X \setminus (A' \cup B')} \lambda_x x + \sum_{x \in B'} (\lambda_x + \lambda) \quad \text{by putting a bit of order} \\
&\qquad \uparrow \\
&\qquad \text{look here}
\end{aligned}
$$

The last line is another linear combination for b, in which all coefficients are positive but $\lambda_x - \lambda = 0$ for some $x \in A'$ by definition of λ. So we found $\tilde{X} \subsetneq X$ such that $b \in \texttt{cone}(\tilde{X})$. We can iterate and remove elements from X until $|X| \leq 2d \log_2(4dM)$. Completely crazy! □

Carathéodory bound given in Lemma 109 can be applied to system of linear equations. Informally, it has few equations and many variables, then if the system is satisfiable, then it has a solution with few non-zero variables.

Corollary 111. *Consider a system of linear equations with d equations and $|var|$ variables with coefficients in $\{-M, -M+1, \ldots, M-1, M\}$.*

Then if it has a integral positive solution, then it has an integral solution where less that $2d \log_2(4dM)$ variables are non-zero.

Proof. The system is of the form $A\vec{s} = \vec{k}$. Suppose it has a integral positive solution $\vec{s} \in \mathbb{N}^{|var|}$.

Let $X =$ columns of matrix A. As $A\vec{s} = \vec{k}$ we have that $\vec{k} \in \texttt{cone}(X)$. By Lemma 109, there is a subset $\tilde{X}$ of the columns of matrix X of size $|\tilde{X}| \leq 2d \log_2(4dM)$ such that $\vec{k} \in \texttt{cone}(\tilde{X})$. Just means that the variable corresponding to missing columns are zeros. □

8.4.2 Application to our setting

Proposition 112. *Suppose that the linear system $\mathcal{S}$ on $\#p_1, \ldots, \#p_d$ has a solution where non-empty regions are in $\mathbb{B}_0 \subseteq \{0,1\}^d$. Then it a solution where non-empty regions is $\mathbb{B} \subseteq \mathbb{B}_0$ such that $|\mathbb{B}| \leq 2d \log_2(4d)$.*

Proof. $\boxed{\Uparrow}$ Obvious.

$\boxed{\Downarrow}$ Consider $k_1, \ldots, k_d$ to be the values of $[\![\#\psi_1]\!]_{G,u}, \ldots, [\![\#\psi_d]\!]_{G,u}$ in the solution G, u of $\mathcal{S}$. To apply Corollary 111, we rewrite $\mathcal{S}$ as the following system with d equations:

$$\begin{cases} k_1 = \sum_{\rho \in \mathbb{B}_0 | \rho_1 = 1} s_\rho \\ \vdots \\ k_d = \sum_{\rho \in \mathbb{B}_0 | \rho_d = 1} s_\rho \end{cases}$$

In a vectorial form, we get:

$$\begin{pmatrix} k_1 \\ \vdots \\ k_d \end{pmatrix} = \sum_{\rho \in \mathbb{B}_0} s_\rho \begin{pmatrix} 1_{\rho_1} \\ \vdots \\ 1_{\rho_d} \end{pmatrix}$$

By Corollary 111, there exists a subset $\mathbb{B} \subseteq \mathbb{B}_0$ of size at most $2d \log_2(4d)$ and values for the s_ρ such that

$$\begin{pmatrix} k_1 \\ \vdots \\ k_d \end{pmatrix} = \sum_{\rho \in \mathbb{B}} s_\rho \begin{pmatrix} 1_{\rho_1} \\ \vdots \\ 1_{\rho_d} \end{pmatrix}.$$

□

8.5 Application: PSPACE Tableau Method for $K^\#$

We write a non-deterministic procedure inspired from [Nunn and Schwarzentruber, 2023] (in which we forgot to use QFBAPA) and [Baader, 2017] (in which QFBAPA is used but for a description logic close to $K^\#$).

8.5.1 Description of the algorithm

input: Γ a finite set of $K^{\#}$-formulas
output: has a non-rejecting execution iff Γ is $K^{\#}$-satisfiable
procedure sat$K^{\#}(\Gamma)$

for non-#-nested expressions 1_φ **do**
choose either to add φ in Γ and replace 1_φ by 1 in Γ
or to add $\neg\varphi$ in Γ and replace 1_φ by 0 in Γ

apply non-deterministically Boolean tableau rules to Γ:

- if $\varphi \wedge \psi \in \Gamma$, remove $\varphi \wedge \psi$ from Γ and add φ and ψ to Γ
- if $\varphi \vee \psi \in \Gamma$, remove $\varphi \vee \psi$ from Γ and choose either to add φ to Γ or add ψ to Γ
- if φ and $\neg\varphi$, clash

let $\#\psi_1, \ldots \#\psi_d$ be a list of all the non-#-nested constructions of the form $\#\psi$ in Γ
choose a subset $\mathbb{B} \subseteq \{0,1\}^d$ of cardinality $\leq 2d\log_2(4d)$
replace each occurrence of $\#\psi_i$ by $\sum_{\rho\in\mathbb{B}|\rho_i=1} s_\rho$ in Γ
let $\mathcal{S}$ be the set of inequalities in Γ
check that $\mathcal{S}$ is ILP-satisfiable
for $\rho \in \mathbb{B}$ **do**
sat$K^{\#}(\{\neg\psi_i \mid \rho_i = 0\} \cup \{\psi_i \mid \rho_i = 1\})$

In the algorithm, at each step, we extract the set of inequalities in Γ. For instance, we may get S to be

$$\begin{cases} \#\psi_1 + 3\#\psi_2 \leq 5 \\ 2\#\psi_1 + 4\#\psi_3 \leq 42 \end{cases}$$

The use we make of QFBAPA is "trivial" since set expressions and set variables coincide: they are $\psi_1, \psi_2, \ldots$. The content of ψ_i may be Boolean (and also modal) but the Boolean reasoning is handled by tableau rules, not by a QFBAPA reasoner. So the use of the QBFPAPA technique is for $d = e$ (we write $\mathbb{B} \subseteq \{0,1\}^e$ instead of $\mathbb{B} \subseteq \{0,1\}^d$).

We then guess $\mathbb{B}$ which are the non-zero regions. For instance, having

$$\mathbb{B} := \{100, 101, 111\}$$

mean that we are trying to create successors for the current vertex where

1. $\psi_1 \wedge \neg\psi_2 \wedge \neg\psi_3$ holds in a subset of successors (s_{100} is the cardinal);
2. $\psi_1 \wedge \neg\psi_2 \wedge \psi_3$ holds in a subset of successors (s_{101} is the cardinal);
3. $\psi_1 \wedge \psi_2 \wedge \psi_3$ holds in a subset of successors (s_{111} is the cardinal).

Other combinations (e.g. $\neg\psi_1 \wedge \psi_2 \wedge \psi_3$) are not present in the model we construct.

We then replace $\#\psi_1$ by the sum of s_{100} (if 100 in $\mathbb{B}$), s_{101} (if 101 in $\mathbb{B}$), s_{110} (if 110 in $\mathbb{B}$), s_{111} (if 111 in $\mathbb{B}$). Similarly for $\#\psi_2$ and $\#\psi_3$.

As we are going then to check for satisfiability of 1.-3. (for loop at the end of the algorithm), we add that $s_\rho \geq 1$ for all non-zero regions ρ.

8.5.2 Soundness and completeness

Proposition 113. *If $\mathsf{sat}K^\#(\Gamma)$ has an accepting execution then Γ is $K^\#$-satisfiable.*

Proof. We consider an accepting execution.

We construct the root to be a vertex u labelled by $\ell_0(u)$ constrained by the presence/absence of $x_i = 1$ in Γ.

By induction we know that for each non-region $\rho \in \mathbb{B}$, we know that $\psi_\rho := \bigwedge_{i|\rho_i=0} \neg\psi_i \wedge \bigwedge_{i|\rho_i=1} \psi_i$ is $\mathrm{K}^\#$-satisfiable in a model G_ρ, u_ρ. As S' is satisfiable, there is a QFPA-model $[\![\cdots]\!]$ for S'.

We make $[\![s_\rho]\!]$ copies of G_ρ, u_ρ that we declare as successors of u. We obtain a pointed graph satisfying Γ. □

Proposition 114. *If Γ is $K^\#$-satisfiable, then $\mathsf{sat}K^\#(\Gamma)$ has an accepting execution.*

Proof. Consider a pointed graph G, u which is a model of Γ. We apply the Boolean tableau rules accordingly for $\vee$, 1_φ (e.g. choosing to put φ in Γ if φ holds in G, u and 0 otherwise). As Γ holds in G, u, $\mathcal{S}$ is QFBA-satisfiable: let $\mathcal{M}$ be a model of $\mathcal{S}$. Let $\mathbb{B}_0$ be the subset of regions that are non-empty in $\mathbb{B}_0$. By Proposition 112, there is a subset $\mathbb{B} \subseteq \mathbb{B}_0$ with $|\mathbb{B}| \leq 2d\log_2(4d)$ such that there is a model $\mathcal{M}'$ in which $\mathcal{S}$ is satisfied and $\mathbb{B}$ is the exactly the set of non-zero regions. We consider the execution that guesses exactly $\mathbb{B}$. As $\mathbb{B} \subseteq \mathbb{B}_0$, we know that ψ_ρ is satisfiable for all $\rho \in \mathbb{B}$. By induction there an accepting execution. □

8.6 Making the exponential blow-up fake

In Proposition 87, we saw that the $\mathrm{K}^\#$-formula φ obtained from a GNN is potentially of exponentially size. This makes our PSPACE algorithm highly inefficient. And then verification is not possible when weights are rational. This is very sad.

Be happy because here is a solution.

8.6.1 Representation the problematic constructions succinctly

First, we should consider the two constructions $\sum_{i=1}^{M^t-1} i \times 1_{E=i} + M^t \times 1_{E\geq M^t}$ and $\sum_{i=1}^{M^t} i \times \#(E=i)$ succinctly. What does it mean? It means we should write $\sum_{i=1}^{M^t-1} i \times 1_{E=i} + M^t \times 1_{E\geq M^t}$ explicitly by expanding the sum. Instead, the sums are 'regular' and we consider them as a new built-in expression $f(M^t, E)$ and $g(M^t, E)$. In the sequel, for clarity, we still continue to write them as $\sum_{i=1}^{M^t-1} i \times 1_{E=i} + M^t \times 1_{E\geq M^t}$ and $\sum_{i=1}^{M^t} i \times \#(E=i)$.

8.6.2 Treating $\sum_{i=1}^{M^t-1} i \times 1_{E=i} + M^t \times 1_{E\geq M^t}$

We handle $\sum_{i=1}^{M^t-1} i \times 1_{E=i} + M^t \times 1_{E\geq M^t}$ as follows. The treatment is somehow easy because at most one term is equal to 1. For each occurrence of the construction $\sum_{i=1}^{M^t-1} i \times 1_{E=i} + M^t \times 1_{E\geq M^t}$:

- choose $i \in \{0, 1, \ldots, M^t\}$
- replace the expression by i
- If $i \in \{1, \ldots, M^t - 1\}$, replace the expression by i and add the constraint $E = i$ in Γ
- If $i = 0$, replace the expression by 0 and add the constraint $E \leq 0$ in Γ
- If $i = M^t$, replace the expression by M^t and add the constraint $E \geq M^t$ in Γ

8.6.3 Treating $\sum_{i=1}^{M^t} i \times \#(E = i)$

Despite $\sum_{i=1}^{M^t} i \times \#(E = i)$ involves an exponential blow-up when expanded, it does not augment the number of equations. However, if we treat each $\#(E = i)$ like a normal $\#\psi$ then d will be exponential. This is terribly bad because choosing $\mathbb{B}$ is of exponential size.

However, we know that formulas $E = 1$, $E = 2$, ..., $E = M^t$ are mutually exclusive. So it is useless to consider a region of the form $E = 1 \wedge E = 2 \wedge \ldots$ which would be empty by design.

The solution is to add $O(\log_2 M^t)$ bits to d to encode the value i telling that $E = i$ in a region ρ. Finally, in the for-loops $\rho \in \mathbb{B}$, if the bits in ρ devoted to an expression $\sum_{i=1}^{M^t} i \times \#(E = i)$ encodes the integer i, then we add $E = i$ as a formula in the subcall $\mathsf{sat}K^{\#}$.

8.7 Quantifier-free Fragment Boolean Algebra with Presburger Arithmetic (*)

This section is optional to be read. A QFBAPA formula is propositional formula where each atom is either an inclusion of sets or equality of sets or linear constraints [Kuncak and Rinard, 2007]. Sets are denoted by Boolean algebra expression, e.g., $(S \cup S') \setminus S''$, or $\mathcal{U}$ where $\mathcal{U}$ denotes the set of all points in some domain. Here S, S', etc. are set variables. Linear constraints are over $|B|$ denoting the cardinality of the set denoted by the set expression B. Let us give a formal definition of the syntax and semantics.

Definition 115. *(see Figure 1 in [Kuncak and Rinard, 2007]) A QFBAPA formula is generated by the axiom φ in the following BNF grammar:*

$$
\begin{array}{l}
\varphi ::= A \;\mid\; \varphi_1 \vee \varphi_2 \;\mid\; \neg\varphi \\
A ::= B_1 = B_2 \;\mid\; B_1 \subseteq B_2 \;\mid\; E_1 = E_2 \;\mid\; E_1 \leq E_2 \\
B ::= S \mid \emptyset \mid \mathcal{U} \mid B_1 \cup B_2 \mid \overline{B} \\
E ::= x \;\mid\; k \;\mid\; E_1 + E_2 \;\mid\; k \times E \;\mid\; |B|
\end{array}
$$

where S ranges in a countable set of set variables, x ranges in a countable set of integer variables, k ranges in $\mathbb{Z}$.

The non-terminal B corresponds to *set expressions* like $\mathit{pianist} \cap \mathit{student}$. Both *pianist* and *student* are set variables. The non-terminal E corresponds to *integer expressions* like $|\mathit{pianist} \cap \mathit{student}| + x - 2$, where x a *integer variable*.

Example 116 (of a QFBAPA formula)**.** $|pianist \cap student| + x \geq 5 \quad \wedge \quad (|pianist| \leq 10 \vee |student| \leq 10)$

The original QFBAPA [Kuncak and Rinard, 2007] also contains the construction k divides E where k is an integer and E an expression. We omit it here since we do not use it. They also have a constant MAXC which is always equal to $|\mathcal{U}|$.

Definition 117. *A QBFBAPA model is a tuple* $\mathcal{M} = (D, [\![\cdots]\!])$ *where*

- D *is a (possible empty) finite set, called the* domain*;*
- *for all integer variables* x, $[\![x]\!] \in \mathbb{Z}$*;*
- *for all set variables* S, $[\![S]\!] \subseteq D$*.*

We naturally extends $[\![\cdots]\!]$ to integer expressions E and set expressions B as follows:

$$\begin{aligned}
[\![k]\!] &:= k \\
[\![E_1 + E_2]\!] &:= [\![E_1]\!] + [\![E_2]\!] \\
[\![k \times E]\!] &:= k \times [\![E]\!] \\
[\![|B|]\!] &:= |[\![B]\!]| \\
[\![B_1 \cup B_2]\!] &:= [\![B_1]\!] \cup [\![B_2]\!] \\
[\![\overline{B}]\!] &:= \overline{[\![B]\!]} \\
[\![\mathcal{U}]\!] &:= D
\end{aligned}$$

Definition 118. *The truth conditions are given as follows:*

$$\begin{aligned}
\mathcal{M} &\models B_1 = B_2 \text{ iff } [\![B_1]\!] = [\![B_2]\!] \\
\mathcal{M} &\models B_1 \subseteq B_2 \text{ iff } [\![B_1]\!] \subseteq [\![B_2]\!] \\
\mathcal{M} &\models E_1 = E_2 \text{ iff } [\![E_1]\!] = [\![E_2]\!] \\
\mathcal{M} &\models E_1 \leq E_2 \text{ iff } [\![E_1]\!] \leq [\![E_2]\!]
\end{aligned}$$

This theorem is left as an exercise.

Theorem 119. *QFBAPA satisfiability problem is in* NP*.*

Open questions

- Efficient implementation
- Nice proof system
- Write down a comprehensive proof for the satisfiability problem of $K^{\#}$ for undirected graphs (and other classes of graphs)

Exercises

Exercise 20. *Take a $K^{\#}$-formula of your choice and apply the algorithm $\mathsf{sat}K^{\#}$.*

Exercise 21. *Explain why if we replace the last part in the naïve algorithm then the algorithm is not correct.*

> *Replace each occurrence of $\#\psi_i$ by $\sum_{\rho\in\{0,1\}^d|\rho_i=1} s_\rho$ in Γ*
> *let $\mathcal{S}$ be the set of inequalities in Γ*
> *check that $\mathcal{S}$ is ILP-satisfiable*
> ***for** all regions $\rho \in \{0,1\}^d$ such that $s_\rho \geq 1$ **do***
> | $\mathsf{sat}K^{\#}(\{\neg\psi_i \mid \rho_i = 0\} \cup \{\psi_i \mid \rho_i = 1\})$

Exercise 22. *Show that QFBAPA is in* NP *using Corollary 111.*

Exercise 23. *Show that QFBAPA is* NP*-hard even if numbers that are written are in formulas are 0 and 1.*

Exercise 24. *Prove formally the theorems of the chapter.*

Exercise 25. *Adapt the algorithm $\mathsf{sat}K^{\#}$ when we are searching for an undirected graph.*

Exercise 26. *What can we say about the complexity/decidability of the extension of QFBAPA where multiplication is allowed, e.g. $|B| \times |B'| + k \times k' \times k'' \geq 2$?*

Exercise 27. *Write the pseudo-code of the algorithm that treats $\sum_{i=1}^{M^t-1} i \times 1_{E=i} + M^t \times 1_{E\geq M^t}$ and $\sum_{i=1}^{M^t} i \times \#(E = i)$ appropriately as explained in Section 8.6.*

Chapter 9

Other settings

In this section, we reproduce some proofs of [Benedikt et al., 2024] with our $\mathrm{K}^{\#}$-formalism, and also using our language of GNN expressions. We also discuss expressivity results with global readout.

9.1 Global readout

We consider now GNNs with global readout, meaning that the update is:

$$\begin{aligned} N^{(t)}(G,u) = \vec{\alpha}(&A_t \times N^{(t-1)}(G,u) \\ &+ B_t \times \sum \{\!\{ N^{(t-1)}(G,v) \mid v \in E(u) \}\!\} \\ &+ C_t \times \sum \{\!\{ N^{(t-1)}(G,v) \mid v \in V \}\!\} \\ &+ b_t). \end{aligned}$$

The term $C_t \times \sum \{\!\{ N^{(t-1)}(G,v) \mid v \in V \}\!\}$ is the global readout term.

9.1.1 Logic $K^{\#,\#^g}$

To capture global readout, we extend $\mathrm{K}^{\#}$ with a universal operator $\#^g\varphi$ which is the global counting in the whole graph whose semantics is

$$[\![\#^g\varphi]\!]_{G,u} := |[\![\varphi]\!]_G|$$

The obtained logic is $K^{\#,\#^g}$. In this logic, we can simulate the *global modality* $\Box^g$ as follows:

$$\Box^g\varphi \quad := \quad \#^g(\varphi) = \#^g(\top)$$

which says that φ holds in all vertices. This logic captures GNNs with global readout in the same way than $\mathrm{K}^{\#}$ captures GNNs. It means that we can restate Proposition 85 and Proposition 87 for $\mathrm{K}^{\#}$ and GNN-expressions with a new construction agg^g for the global readout aggregation.

9.1.2 Verification tasks

Theorem 120. *(see Theorem 23 in [Chernobrovkin et al., 2025]) The satisfiability problem of $K^{\#,\#^g}$ in directed graphs is* NEXPTIME*-complete.*

Theorem 121. *(adapted from Theorem 4.16, [Benedikt et al., 2024]) The satisfiability problem of $K^{\#,\#^g}$ in undirected graphs is undecidable.*

Proof. We recall Hilbert's tenth problem [Matiyasevich, 2005]. The following problem is undecidable:

- input: a set $\mathcal{E}$ of equations of the form $x = 1$, $x = x' + x''$, or $x = x' \times x''$;
- output: yes if $\mathcal{E}$ has a solution in $\mathbb{N}$.

Example 122. *Here is an example of an instance of Hilbert's tenth problem:*

$$\begin{aligned} x_1 &= 1 \\ x_2 &= 1 \\ x_3 &= x_1 + x_2 \text{ meaning that } x_3 = 2 \\ x_5 &= x_4 \times x_3 \text{ meaning } x_5 = 2 \times x_4 \\ x_6 &= x_4 \times x_4 \\ x_6 &= x_5 + x_1 \text{ meaning } x_4^2 = 2x_4 + 1 \end{aligned}$$

It has the solution: $(x_1, x_2, x_3, x_4, x_5, x_6) = (1, 1, 2, 1, 2, 1)$.

We reduce from the Hilbert's tenth problem. We are going to build a $K^{\#,\#^g}$-formula $tr(\mathcal{E})$ in poly-time in the number of bits to represent $\mathcal{E}$ such that $\mathcal{E}$ has a solution in $\mathbb{N}$ iff $tr(\mathcal{E})$ is satisfiable.

Intuition. The idea is to constrain a set (a graph!) where vertices are tagged by equations *and* variables. The subset of vertices tagged by equation e and variable x is denoted by $V_{e,x}$. The $V_{e,x}$ should be a partition of the set V of all vertices:

$$V = \bigsqcup_{e \in \mathcal{E}, \text{variables } x} V_{e,x}.$$

More importantly, for all x, $|V_{e,x}|$ should not depend on e and should be the value of x in a solution of $\mathcal{E}$! The subset $V_{e,x}$ are indexed by e in order to handle the constrain of each equation e separately.

We introduce the following propositions: p_e that says that a vertex is tagged by equation e, and p_x that says that a vertex is tagged by variable x. Intuitively, vertices in $V_{e,x}$ are those in which both p_e and p_x hold.

We now define $tr(\mathcal{E})$ that does the job. First:

$$\bigwedge_{e,e' | e \neq e'} \Box^g \neg (p_e \wedge p_{e'}) \tag{9.1}$$

$$\bigwedge_{x,x' | x \neq x'} \Box^g \neg (p_x \wedge p_{x'}) \tag{9.2}$$

$$\bigwedge_x \bigwedge_{e,e'} \#^g (p_e \wedge p_x) = \#^g (p_{e'} \wedge p_x) \tag{9.3}$$

Up to now, at most one equation tag, at most one variable tag and $|V_{e,x}|$ only depend on x. In order to emphasize that point, we write $|V_{\bullet,x}|$ instead of $|V_{e,x}|$.

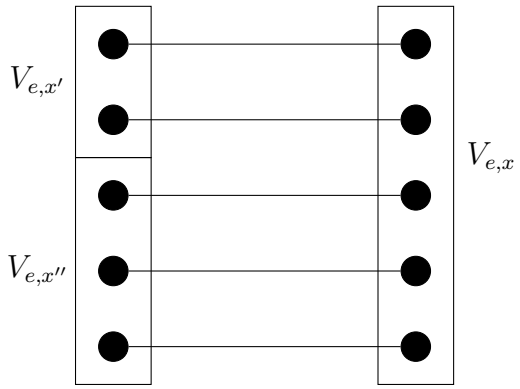
We now introduce a formula φ_e to simulate the semantics of each equation e. For all equations e.

- $\boxed{e = \ x = 1}$ We set $\varphi_e := \#^g(p_e \wedge p_x) = 1$ to intuitively say $|V_{e,x}| = |V_{\bullet,x}| = 1$.

- $\boxed{e = \ x = x' + x''}$ We set

$$\varphi_e := \Box^g \left(\begin{array}{l} ([p_e \wedge (p_{x'} \vee p_{x''})] \to \#(p_e \wedge p_x) = 1) \wedge \\ ([p_e \wedge p_x] \to \#(p_e \wedge (p_{x'} \vee p_{x''})) = 1) \end{array} \right)$$

The formula φ_e says that each vertex in $V_{e,x'} \sqcup V_{e,x''}$ has a single neighbour in $V_{e,x}$. Conversely, each vertex in $V_{e,x}$ has a single neighbour in $V_{e,x'} \sqcup V_{e,x''}$. Said differently, there is a bijection between $V_{e,x'} \sqcup V_{e,x''}$ and $V_{e,x}$.



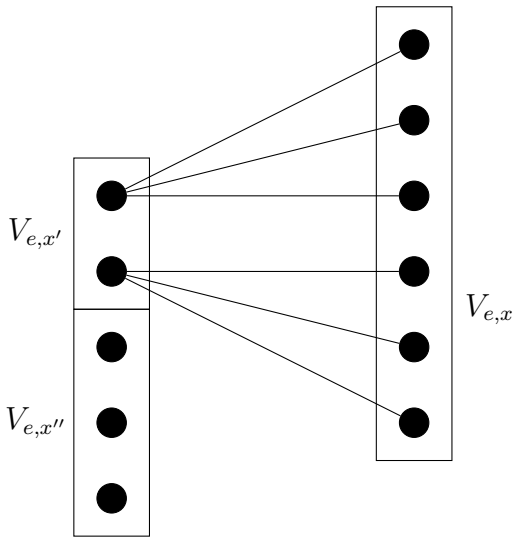
$5 = 2 \times 3$

So $|V_{\bullet,x}| = |V_{\bullet,x'}| + |V_{\bullet,x''}|$ which appropriately simulate the equation $x = x' + x''$.

- $\boxed{e = \ x = x' \times x''}$ We set

$$\varphi_e := \Box^g \left(\begin{array}{l} ([p_e \wedge p_{x'} \to \#(p_e \wedge p_x) = \#^g(p_e \wedge p_{x''})] \wedge \\ ([p_e \wedge p_x] \to \#(p_e \wedge p_{x'}) = 1) \end{array} \right)$$

The formula φ_e says that each vertex in $V_{e,x'}$ has $|V_{e,x''}|$ neighbours in $V_{e,x}$. Conversely, each vertex in $V_{e,x}$ has one neighbour in $V_{e,x'}$. So there is a $|V_{e,x''}|$-to-one mapping from $V_{e,x'}$ to $V_{e,x}$.



$6 = 2 \times 3$

So $|V_{\bullet,x}| = |V_{\bullet,x'}| \times |V_{\bullet,x''}|$ which appropriately simulate the equation $x = x' \times x''$.

The reduction is given by:

$$tr(\mathcal{E}) := (7.1) \wedge (7.2) \wedge (7.3) \wedge \bigwedge_{e \in \mathcal{E}} \varphi_e.$$

It is left to prove formally that $\mathcal{E}$ has a solution in $\mathbb{N}$ iff $tr(\mathcal{E})$ is satisfiable. We left the end of the proof to the reader. □

Corollary 123. *The satisfiability problem of a GNN with TruncReLU as an activation function and with global readout is undecidable.*

Proof. By the previous and a similar translation from $K^{\#,\#^g}$ into GNNs with TruncReLU and global readout. □

9.2 ReLU

Theorem 124. *Verification of GNNs with global readout and* ReLU *as an activation function on undirected graphs is undecidable.*

Proof. Because we can simulate truncReLU with ReLU. It is was already undecidable for truncReLU. □

Theorem 125. *Verification of GNNs with global readout and* ReLU *as an activation function on directed graphs is undecidable.*

Proof. Reduction from *Hilbert's tenth problem*. We still use the fact truncReLU can be simulated by ReLU: this enables to express logical properties. But we also need ReLU for storing arbitrary long integers throw the computation.

The idea is to tag the vertices with a feature $count_x$. When $count_x = 0$ the vertex does not count and $count_x = 1$ it means that the vertex counts for 1 in the value of variable x. We interpret the value of x by $\#^g count_x$. More precisely:

$$\#^g count_x := agg^g(\mathsf{ReLU}(count_x)).$$

- $\boxed{e = \ x = 1}$ $\varphi_e := \#^g count_x = 1$. We can write equalities by inequalities and write inequalities as for the translation from $\mathrm{K}^{\#}$ into GNN with truncReLU (and we simulate truncReLU by ReLU).

- $\boxed{e = \ x = x' + x''}$ $\varphi_e := \#^g count_x = \#^g count_{x'} + \#^g count_{x''}$.

- $\boxed{e = \ x = x' \times x''}$ We cannot write multiplication constraint directly. So introduce extra features y_e and y''_e.

$$\varphi_e := ((y_e = 0) \wedge (z_e = 0)) \vee ((y_e = \#^g count_{x'}) \wedge (z_e = 1)) \tag{9.4}$$
$$\wedge\, agg^g(y_e) = \#^g count_x \tag{9.5}$$
$$\wedge\, agg^g(z_e) = \#^g count_{x''} \tag{9.6}$$

The objective of (7.4) is to replace each 1-value of z_e by $\#^g count_{x'}$ so that $y_e = \#^g count_{x'} \times z_e$ at each vertex. By taking the global aggregation over all vertices, we get:

$$\#^g count_x = \#^g count_{x'} \times \#^g count_{x''}.$$

Finally the GNN is: $\bigwedge_{e \in \mathcal{E}} \varphi_e$. We left the equivalence to the reader. □

Theorem 126. *Verification of GNNs with ReLU as an activation function on undirected graphs is undecidable.*

Proof. We simulate the global modality by a local one around the root. □

9.3 Global readout Modulo FO

We saw that AC-GNN $\cap$ FO = GML, i.e. when there is no global readout. However, it turns out that:

$$\mathsf{FOC}_2 \subsetneq \text{ACR-GNN} \cap \mathsf{FO}$$

The seminar result in that direction is in [Hauke and Walega, 2026]. We prove that the inclusion is strict by considering the property Strict linear order: 'the relation of the graph is a strict linear order'.

Theorem 127. *Strict linear order is expressible by some ACR-GNN with some complicated aggregation functions... [Hauke and Walega, 2026].*

We actually have:

Theorem 128. *Strict linear order is expressible by some ACR-GNN with max and sum as aggregation functions and* ReLU *for the activation function [Funk and Kojelis, 2026].*

Strict linear order expressible in FO

Proposition 129. *Strict linear order is expressible in FO.*

Proof. The conjunction of the three formulas expresses Strict linear order:

$$\forall x \neg R(x,x)$$
$$\forall x, \forall y, (x = y) \vee R(x,y) \vee R(y,x)$$
$$\forall x, \forall y, \forall y, [R(x,y) \wedge R(y,z)] \rightarrow R(x,z)$$

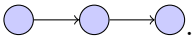
□

Strict linear order not expressible in FOC_2

Proposition 130. FOC_2 *cannot express Strict linear order.*

Proof. The proof in [Hauke and Walega, 2026] is based on a similar characterisation than Theorem 70 of expressivity of a formula in FOC_2 with a quantifier-depth r and 'counting ability', i.e. a bound on the numbers in the formula. □

A GNN for Strict linear order

Let P_2 be the path of length 2, i.e. the graph  .

The following lemma is a characterisation of the property Strict linear order involving quantities that can be computed by GNNs.

Lemma 131. *[Funk and Kojelis, 2026] A graph G is a strict linear order if $|E| = \binom{|V|}{2}$ and $|hom(P_2, G)| = \binom{|V|}{3}$ where $|hom(H,G)|$ is the number of homomorphism from H to G, i.e. here, the number of occurrences of occurrences of P_2 in G.*

We define a GNN of the form $\mathsf{ReLU}((\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4)) \leq 0$ where $\vartheta_1 = |E| - \binom{|V|}{2}, \vartheta_2 = \binom{|V|}{2} - |E|, \vartheta_3 = |hom(P_2,G)| - \binom{|V|}{3}$ and $\vartheta_4 = \binom{|V|}{3} - |hom(P_2,G)|$.

- We compute $|V|$ with $\mathsf{ReLU}(agg^g(1))$.
- We compute $|E|$ with $\mathsf{ReLU}(agg^g(\mathsf{ReLU}(agg(1)))$.
- We compute $|V|^2$ with $\mathsf{ReLU}(agg^g(|V|)$.
- We compute $|V|^3$ with $\mathsf{ReLU}(agg^g(|V|^2)$.
- We compute $\binom{|V|}{2}$ with $\mathsf{ReLU}(-\frac{1}{2}|V| + \frac{1}{2}|V|^2)$
- We compute $\binom{|V|}{3} = \frac{|V|(|V|-1)(|V|-2)}{6}$ with $\mathsf{ReLU}(\frac{1}{6}|V|^3 - 1/2 \times |V|^2 + 1/3|V|)$
- We compute at each vertex the number of paths of length 2 with

$$nbP2 = \mathsf{ReLU}(agg(\mathsf{ReLU}(agg(1))).$$

- We compute $|hom(P_2,G)|$ with $agg^g(nbP2)$.

9.4 Max as an aggregation function

The aggregation we used mainly was the sum. When the aggregation function is max then ACGNNs has the same expressive power than ML. In this setting, Cucala et al. [Cucala and Grau, 2024] uses Datalog as a formalism to explain classification of GNNs.

9.5 GNNs with random initialisation

A ACR-GNN with random initialisation (we write riACRGNN) is a *randomized algorithm* that:

- extend the feature vectors of each vertex with a random value
- perform the execution a standard GNN as usual

Theorem 132. *[Abboud et al., 2021] Let $n \geq 1$. Consider a function $f : \{$graphs with n vertices$\} \to \mathbb{R}$. Then for all $\epsilon, \delta > 0$, there exists a riACRGNNN such that for all graphs G with n vertices, we have:*

$$\mathbb{P}(|f(G) - N(G)| < \epsilon) \geq 1 - \delta.$$

9.6 Descriptive complexity and GNNs (*)

In this section, we sum up the work in [Grohe, 2024].

9.6.1 Family of GNNs

A family of $GNNs$ is an infinite sequence $N_1, N_2, N_3, \ldots$ of GNNs. The role of the GNN N_n is to be applied to graphs with n vertices. The weight of a GNN N is the sum number of computation nodes of the underlying neural networks plus the sum of the absolute values of all parameters of these networks. A N is bounded-depth if the number of layers is fixed (it is almost assumed everywhere in this lecture). A rational piecewise linear (rpl) function is a function that is a 'concatenation' of linear functions where the 'junctions' occurs at rational points.

Example 133. $\mathsf{truncReLU}$, ReLU, *etc.*

A rpl-approximable activation function is a function that can be approximated by a sequence of rpls.

Example 134. *sigmoid...*

9.6.2 $FO + C$

$FO + C$ is stronger than FOC, and extends $F0$ with counting terms as follows.

$$\vartheta, \vartheta', \ldots ::= 0 \mid 1 \mid \vartheta + \vartheta' \mid \vartheta \times \vartheta' \mid \#(x_1, \ldots, x_k, y_1 < \vartheta_1, \ldots, y_k < \vartheta_k).\varphi$$
$$\varphi, \psi, \ldots ::= x_1 = x_2 \mid \vartheta \leq \vartheta' \mid R(x, y) \mid \neg\varphi \mid \varphi \vee \psi$$

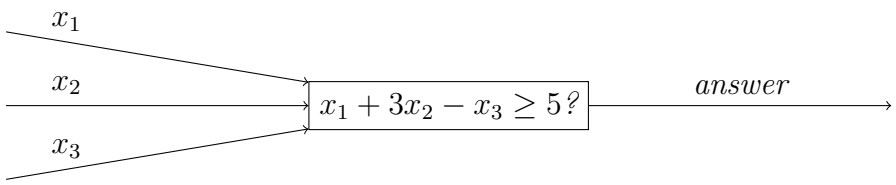
Intuitively, $\#(x_1, \ldots, x_k, y_1 < \vartheta_1, \ldots, y_k < \vartheta_k).\varphi$ is the number of tuples $(x_1, \ldots, x_k, y_1, \ldots, y_k)$ where $x_1, \ldots, x_k$ are vertices, $y_1, \ldots, y_k$ are non-negative numbers such that

$$y_1 < \vartheta_1 \wedge \cdots \wedge y_k < \vartheta_k \wedge \varphi.$$

9.6.3 TC^0

TC^0 is the class of decision problems that are computable by families of bounded-depth polynomial-size Boolean circuits with threshold gates, which are like a linear inequation.

Example 135. *Here is an example of a threshold gate. It takes three Boolean inputs $x_1, x_2, x_3 \in \{0, 1\}$ and outputs 1 if $x_1 + 3x_2 - x_3 \geq 5$ and 0 otherwise.*



The following theorem concerns a vertex query that is invariant up to isomorphism.

Theorem 136. *(see Th. 8.6 in [Grohe, 2024]) A vertex query is computable by a poly-weight family of GNNs with rpl-approximable activation functions*

iff

it is expressible in $GFO + C_{nu}$, the guarded fragment of FO extended with arbitrary relations on non-negative integers.

implies it is in TC_0.

Theorem 137. *A vertex query is computable by a poly-weight family of riACRGNNs, with rpl-approximable activation functions*

iff it is in TC_0.

9.7 Transformers

There are essentially two models that use transformers as a subroutine: graph transformer networks (GT) and General, Powerful, Scalable graph transformer networks (GPS). The correspondence with some logic have been studied in [Ahvonen et al., 2026].

A *graph transformer network* (GT) [Yun et al., 2019] is an algorithm like a GNN but each layer a global transformer readout operation. A General, Powerful, Scalable graph transformer network GPS-network [Rampásek et al., 2022] is an algorithm where global attention mechanism and message-passing (combine-aggregate) layers alternate.

Theorem 138. *[Ahvonen et al., 2026] We have:*

Propositional logic + non-counting global modality	$= GT \cap \mathsf{FO}$
GML *+ non-counting global modality*	$= GPS \cap \mathsf{FO}$
Propositional logic + counting global modality	$=$ *GT on arbitrary large floating numbers*
GML *+ counting global modality*	$=$ *GPS on arbitrary large floating numbers*

9.8 Recurrent GNNs

In a recurrent GNN, a layer is repeated until a certain halting condition is satisfied. Pflueger et al. [Pflueger et al., 2024] provides some characterisation in terms of logics with fix-points.

Results in [Ahvonen et al., 2024] are the following. They consider *graded modal substitution calculus* (GMSC) which extends *modal substitution calculus* (MSC) [Kuusisto, 2013]. In both logics, properties are expressed as programs.

Example 139. *The MSC program*

$X(0) : -p \qquad\qquad X : -\Diamond X$

is the vertex property $\varphi(x) :=$ 'there is a path from x to a p-vertex'. The left-hand side is the termination condition, while the right-hand side is the iteration process.

The semantics is as follows:

$G, u \models$ *program* if there exists $n \in \mathbb{N}$ iff G, u satisfies the n-iterated rewriting of the axiom variable of the program, for some $n \in \mathbb{N}$.

Example 140. *The MSC program*

$X(0) : -\Box\bot \qquad\qquad X : -\Diamond X \wedge \Box X$

is the vertex property $\varphi(x) :=$ 'there exists $n \in \mathbb{N}$ such that all paths of length n leads to a vertex with no out-neighbours'. This property is not expressible in monadic second order logic *(MSO).*

MSC includes the fragment of μ-calculus that does not involve ν. The satisfiability problem of both MSC and GMSC are undecidable.

They also consider ω-GML which GML with infinitary version of GML where we can write arbitrary infinite conjunctions. Any GMSC program can be expressed in ω-GML. Logic ω-GML can express undecidable properties while GMSC cannot.

Theorem 141. *[Ahvonen et al., 2024] We have:*

GMSC	$=$ *recurrent GNNs on arbitrary large floating numbers*
ω-GML	$=$ *recurrent GNNs on real numbers*

9.9 Future Work

- Build an implementation
- Probabilistic reasoning
- Learnability VS expressivity
- Build a benchmark for verification in several application domains

Bibliography

[Abboud et al., 2021] Abboud, R., Ceylan, İ. İ., Grohe, M., and Lukasiewicz, T. (2021). The surprising power of graph neural networks with random node initialization. In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 2112–2118. ijcai.org.

[Ahvonen et al., 2026] Ahvonen, V., Funk, M., Heiman, D., Kuusisto, A., and Lutz, C. (2026). Expressive power of graph transformers via logic. In Koenig, S., Jenkins, C., and Taylor, M. E., editors, *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 19569–19579. AAAI Press.

[Ahvonen et al., 2024] Ahvonen, V., Heiman, D., Kuusisto, A., and Lutz, C. (2024). Logical characterizations of recurrent graph neural networks with reals and floats. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C., editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

[Arvind et al., 2015] Arvind, V., Köbler, J., Rattan, G., and Verbitsky, O. (2015). On the power of color refinement. In Kosowski, A. and Walukiewicz, I., editors, *Fundamentals of Computation Theory - 20th International Symposium, FCT 2015, Gdańsk, Poland, August 17-19, 2015, Proceedings*, volume 9210 of *Lecture Notes in Computer Science*, pages 339–350. Springer.

[Baader, 2017] Baader, F. (2017). A new description logic with set constraints and cardinality constraints on role successors. In Dixon, C. and Finger, M., editors, *Frontiers of Combining Systems - 11th International Symposium, FroCoS 2017, Brasília, Brazil, September 27-29, 2017, Proceedings*, volume 10483 of *Lecture Notes in Computer Science*, pages 43–59. Springer.

[Babai, 2016] Babai, L. (2016). Graph isomorphism in quasipolynomial time [extended abstract]. In Wichs, D. and Mansour, Y., editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM.

[Babai et al., 1980] Babai, L., Erdös, P., and Selkow, S. M. (1980). Random graph isomorphism. *SIAM J. Comput.*, 9(3):628–635.

[Barceló et al., 2020] Barceló, P., Kostylev, E. V., Monet, M., Pérez, J., Reutter, J. L., and Silva, J. P. (2020). The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[Barceló et al., 2026] Barceló, P., Geerts, F., Lanzinger, M., Pakhomenko, K., and den Bussche, J. V. (2026). A logical view of gnn-style computation and the role of activation functions.

[Benedikt et al., 2024] Benedikt, M., Lu, C., Motik, B., and Tan, T. (2024). Decidability of graph neural networks via logical characterizations. In Bringmann, K., Grohe, M., Puppis, G., and Svensson, O., editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 127:1–127:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[Berkholz et al., 2017] Berkholz, C., Bonsma, P. S., and Grohe, M. (2017). Tight lower and upper bounds for the complexity of canonical colour refinement. *Theory Comput. Syst.*, 60(4):581–614.

[Blackburn et al., 2001] Blackburn, P., de Rijke, M., and Venema, Y. (2001). *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

[Briglia et al., 2025] Briglia, G., Fabiano, F., and Mariani, S. (2025). Scaling multi-agent epistemic planning through gnn-derived heuristics. *CoRR*, abs/2508.12840.

[Cai et al., 1992] Cai, J., Fürer, M., and Immerman, N. (1992). An optimal lower bound on the number of variables for graph identification. *Comb.*, 12(4):389–410.

[Cardon and Crochemore, 1982] Cardon, A. and Crochemore, M. (1982). Partitioning a graph in o(|a| log2 |v|). *Theor. Comput. Sci.*, 19:85–98.

[Chagrov and Rybakov, 2002] Chagrov, A. V. and Rybakov, M. N. (2002). How many variables does one need to prove pspace-hardness of modal logics. In Balbiani, P., Suzuki, N., Wolter, F., and Zakharyaschev, M., editors, *Advances in Modal Logic 4, papers from the fourth conference on "Advances in Modal logic," held in Toulouse, France, 30 September - 2 October 2002*, pages 71–82. King's College Publications.

[Chernobrovkin et al., 2025] Chernobrovkin, A., Sälzer, M., Schwarzentruber, F., and Troquard, N. (2025). Verifying graph neural networks with readout is intractable.

[Cucala and Grau, 2024] Cucala, D. J. T. and Grau, B. C. (2024). Bridging max graph neural networks and datalog with negation. In Marquis, P., Ortiz, M., and Pagnucco, M., editors, *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2-8, 2024*.

[Ehrenfeucht, 1961] Ehrenfeucht, A. (1961). An application of games to the completeness problem for formalized theories. *Fundamenta Mathematicae*, 49(2):129–141.

[Eisenbrand and Shmonin, 2006] Eisenbrand, F. and Shmonin, G. (2006). Carathéodory bounds for integer cones. *Oper. Res. Lett.*, 34(5):564–568.

[Fraïssé, 1954] Fraïssé, R. (1954). Sur l'extension aux relations de quelques propriétés des ordres. *Annales scientifiques de l'École Normale Supérieure*, 71(4):363–388.

[Funk and Kojelis, 2026] Funk, M. and Kojelis, D. (2026). Towards understanding the expressive power of gnns with global readout. *CoRR*, abs/2604.22870.

[Grohe, 2021] Grohe, M. (2021). The logic of graph neural networks. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–17. IEEE.

[Grohe, 2024] Grohe, M. (2024). The descriptive complexity of graph neural networks. *TheoretiCS*, 3.

[Haase and Zetzsche, 2019] Haase, C. and Zetzsche, G. (2019). Presburger arithmetic with stars, rational subsets of graph groups, and nested zero tests. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–14. IEEE.

[Hauke and Walega, 2026] Hauke, S. P. and Walega, P. A. (2026). Aggregate-combine-readout gnns can express logical classifiers beyond the logic C2. In Koenig, S., Jenkins, C., and Taylor, M. E., editors, *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 21594–21601. AAAI Press.

[Horrocks et al., 2007] Horrocks, I., Hustadt, U., Sattler, U., and Schmidt, R. A. (2007). Computational modal logic. In Blackburn, P., van Benthem, J. F. A. K., and Wolter, F., editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 181–245. North-Holland.

[Huang and Villar, 2021] Huang, N. T. and Villar, S. (2021). A short tutorial on the weisfeiler-lehman test and its variants. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pages 8533–8537. IEEE.

[Immerman and Lander, 1990] Immerman, N. and Lander, E. (1990). Describing graphs: A first-order approach to graph canonization. In *Complexity Theory Retrospective: In Honor of Juris Hartmanis on the Occasion of His Sixtieth Birthday, July 5, 1988*, pages 59–81. Springer.

[Karystinaios et al., 2023] Karystinaios, E., Foscarin, F., and Widmer, G. (2023). Musical voice separation as link prediction: Modeling a musical perception task as a multi-trajectory tracking problem. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 3866–3874. ijcai.org.

[Kuncak and Rinard, 2007] Kuncak, V. and Rinard, M. (2007). Towards efficient satisfiability checking for boolean algebra with presburger arithmetic. In Pfenning, F., editor, *Automated Deduction – CADE-21*, pages 215–230, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Kuusisto, 2013] Kuusisto, A. (2013). Modal logic and distributed message passing automata. In Rocca, S. R. D., editor, *Computer Science Logic 2013, CSL 2013, Torino, Italy, September 2-5, 2013*, volume 23 of *LIPIcs*, pages 452–468. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[Matiyasevich, 2005] Matiyasevich, Y. V. (2005). Hilbert's tenth problem and paradigms of computation. In Cooper, S. B., Löwe, B., and Torenvliet, L., editors, *New Computational Paradigms, First Conference on Computability in Europe, CiE 2005, Amsterdam, The Netherlands, June 8-12, 2005, Proceedings*, volume 3526 of *Lecture Notes in Computer Science*, pages 310–321. Springer.

[Morgan, 1965] Morgan, H. L. (1965). The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of chemical documentation*, 5(2):107–113.

[Morris et al., 2023] Morris, C., Lipman, Y., Maron, H., Rieck, B., Kriege, N. M., Grohe, M., Fey, M., and Borgwardt, K. M. (2023). Weisfeiler and leman go machine learning: The story so far. *J. Mach. Learn. Res.*, 24:333:1–333:59.

[Morris et al., 2019] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4602–4609. AAAI Press.

[Nunn et al., 2024] Nunn, P., Sälzer, M., Schwarzentruber, F., and Troquard, N. (2024). A logic for reasoning about aggregate-combine graph neural networks. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 3532–3540. ijcai.org.

[Nunn and Schwarzentruber, 2023] Nunn, P. and Schwarzentruber, F. (2023). A modal logic for explaining some graph neural networks. *CoRR*, abs/2307.05150.

[Paige and Tarjan, 1987] Paige, R. and Tarjan, R. E. (1987). Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989.

[Pflueger et al., 2024] Pflueger, M., Cucala, D. T., and Kostylev, E. V. (2024). Recurrent graph neural networks and their connections to bisimulation and logic. In Wooldridge, M. J., Dy, J. G., and Natarajan, S., editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pages 14608–14616. AAAI Press.

[Pratt-Hartmann, 2009] Pratt-Hartmann, I. (2009). Logics with counting. Course notes for ESSLLI 2009.

[Rampásek et al., 2022] Rampásek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. (2022). Recipe for a general, powerful, scalable graph transformer. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

[Reiser et al., 2022] Reiser, P., Neubert, M., Eberhard, A., Torresi, L., Zhou, C., Shao, C., Metni, H., van Hoesel, C., Schopmans, H., Sommer, T., and Friederich, P. (2022). Graph neural networks for materials science and chemistry. *Communications Materials*, 3(93).

[Salamat et al., 2021] Salamat, A., Luo, X., and Jafari, A. (2021). Heterographrec: A heterogeneous graph-based neural networks for social recommendations. *Knowl. Based Syst.*, 217:106817.

[Sälzer et al., 2025] Sälzer, M., Schwarzentruber, F., and Troquard, N. (2025). Verifying quantized graph neural networks is pspace-complete. *CoRR*, abs/2502.16244.

[Sato, 2020] Sato, R. (2020). A survey on the expressive power of graph neural networks. *CoRR*, abs/2003.04078.

[Scarselli et al., 2009] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80.

[Schönherr and Lutz, 2026] Schönherr, M. and Lutz, C. (2026). Logical characterizations of gnns with mean aggregation. In Koenig, S., Jenkins, C., and Taylor, M. E., editors, *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 25218–25225. AAAI Press.

[Schöning, 1988] Schöning, U. (1988). Graph isomorphism is in the low hierarchy. *J. Comput. Syst. Sci.*, 37(3):312–323.

[Soeteman et al., 2026] Soeteman, A., Benedikt, M., Grohe, M., and ten Cate, B. (2026). How expressive are graph neural networks in the presence of node identifiers?

[Tobies, 1999] Tobies, S. (1999). A pspace algorithm for graded modal logic. In Ganzinger, H., editor, *Automated Deduction - CADE-16, 16th International Conference on Automated Deduction, Trento, Italy, July 7-10, 1999, Proceedings*, volume 1632 of *Lecture Notes in Computer Science*, pages 52–66. Springer.

[Turing, 1937] Turing, A. M. (1937). On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, s2-42(1):230–265.

[Wittig et al., 2026] Wittig, S., Vasileiou, A., Nerem, R. R., Stoll, T., Geerts, F., Wang, Y., and Morris, C. (2026). Which algorithms can graph neural networks learn? *CoRR*, abs/2602.13106.

[Xiong et al., 2021] Xiong, J., Xiong, Z., Chen, K., Jiang, H., and Zheng, M. (2021). Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6):1382–1393.

[Xu et al., 2019] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[Ye et al., 2022] Ye, Z., Kumar, Y. J., Sing, G. O., Song, F., and Wang, J. (2022). A comprehensive survey of graph neural networks for knowledge graphs. *IEEE Access*, 10:75729–75741.

[Yun et al., 2019] Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. (2019). Graph transformer networks. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11960–11970.

Index