Graph neural networks and logic

François Schwarzentruber
ENS de Lyon

September 29, 2025

Contents

Chapter 1

Graph neural networks and Weisfeiler-Lehman tests

Key reference: [Sat20], [Gro21]

1.1 Motivation

Graph neural networks are used in various applications:

- recommendation in social network [SLJ21],
- knowledge graphs [YKS+22],
- chemistry [RNE+22],
- drug discovery [XXC+21], 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?

1.2 Graph neural networks

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 $\ell_0 : V \to \{0, 1\}^d$.

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

A GNN N is usually defined as a tuple of parameters (see [BKM+20], [NSST24], [BLMT24]). To make it more concrete, we present it as an algorithm (parametrized by the weights computed during some learning process), see Figure 1.1. 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 1.1: 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 1 (graph neural network)**.** *A GNN is an algorithm as shown in Figure 1.1.*

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:

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

1.3 GNN on graphs

For graph classification, the last operation could be:

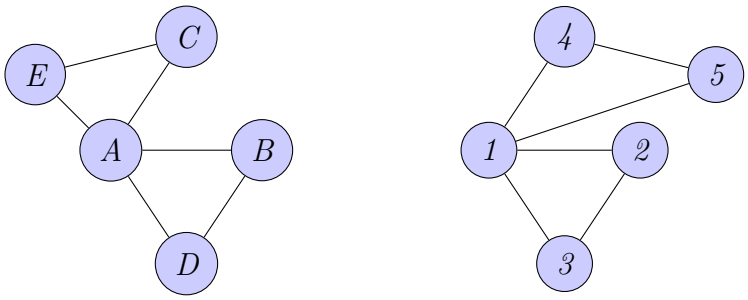
$$\textbf{return yes if } w \sum_{u \in V} \ell_L(u) + b \geq 0 \textbf{ else no}$$

The sum is taken over *all* vertices u in the graph. This kind of operation is *global readout*.

1.4 Graph isomorphism

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

Example 1. *Here are two graphs that isomorphic:*



Definition 2. *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 [Sch88]. Testing graph isomorphisms can be done in quasipolynomial time [Bab16]: 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[2] which do "almost" solve graph isomorphism.

1.5 1-WL aka Color refinement

1.5.1 Description

There are different presentations of 1-WL (1-Weisfeiler-Lehman) aka color refinement aka naive vertex refinement in the literature. 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.

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

[2]It is possible to implement it in $O((|V| + |E|) \log |V|)$ [CC82].

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 1.2: Colour refinement algorithm.

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':

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

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

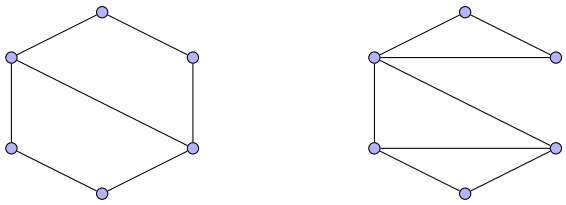
Definition 3. *Two labellings ℓ and ℓ' are equivalent if*

for all vertices $u, v \in V$, $\ell[u] = \ell[v]$ iff $\ell'[u] = \ell'[v]$.

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 1.2 gives the pseudo-code for colour refinement.

1.5.2 Example

Exercise 1. *(from [Gro21]) Compute $\mathsf{cr}(G)$ and $\mathsf{cr}(G')$ for the graphs G and G' below:*



1.5.3 Implementation

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

🎓Read [CC82], also [PT87].

The interested reader may also look at [BBG17] for tight complexity.

1.5.4 Indistinguishability

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 color are the same via $\mathsf{cr}(G)$ and $\mathsf{cr}(G')$. More formally, we define:

$$\{\!\{\mathsf{cr}(G)\}\!\} := \{\!\{\mathsf{cr}(G, u) \mid u \in V\}\!\}$$

Definition 4. *G, G' are cr-indistinguishable if $\{\!\{\mathsf{cr}(G)\}\!\} = \{\!\{\mathsf{cr}(G')\}\!\}$.*

Definition 5. *G, u and G', u' are cr-indistinguishable if $\mathsf{cr}(G, u) = \mathsf{cr}(G', u')$.*

1.5.5 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 2. *If G and G' are isomorphic **then** G, G' are cr-indistinguable.*

Proof. Suppose that G and G' are isomorphic. Let $\ell_0, \ell_1, \ldots$ be the labellings of the execution of $\mathsf{cr}(G)$. Let $\ell'_0, \ell'_1, \ldots$ be the labellings of the execution of $\mathsf{cr}(G')$.

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

$$\text{for all } u \in V,\ \ell_t(u) = \ell'_t(\pi(u)).$$

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

$$\begin{aligned}
\ell_{t+1}[u] &:= (\ell_t[u], \{\!\{\ell_t[v] \mid v \in E(u)\}\!\}) \\
&= (\ell'_t[\pi(u)], \{\!\{\ell_t[\pi(v)] \mid v \in E(u)\}\!\}) \\
&= (\ell'_t[\pi(u)], \{\!\{\ell_t[\pi(v)] \mid \pi(v) \in E'(\pi(u))\}\!\}) \\
&= (\ell'_t[\pi(u)], \{\!\{\ell_t[v'] \mid v' \in E'(\pi(u))\}\!\}) \\
&= \ell'_{t+1}(\pi(u))
\end{aligned}$$

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 $\mathsf{cr}(G')$.
- $\{\!\{\mathsf{cr}(\ell_t)\}\!\} = \{\!\{\mathsf{cr}(\ell'_t)\}\!\}$.

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

The algorithm cr does not characterize isomorphism, as shown in Figure 1.3.

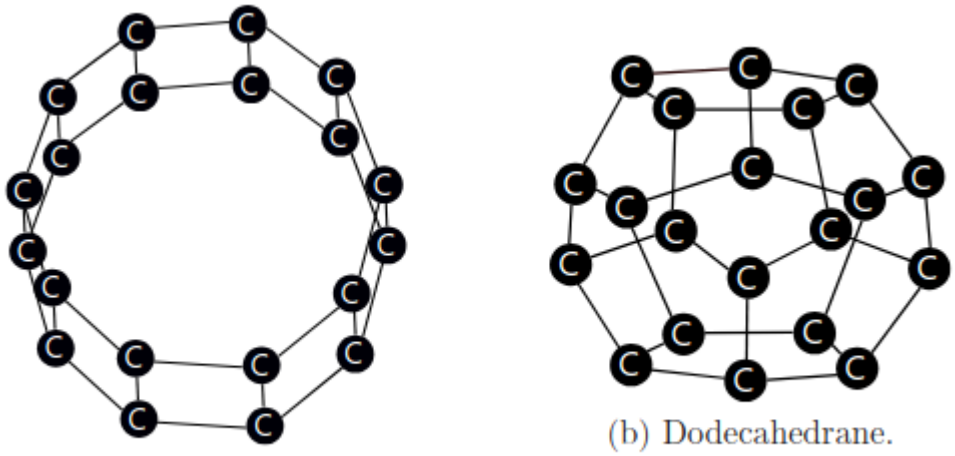
Proposition 3. *[IL90] Let G and G' be two trees.*
G, G' are isomorphic iff G, G' are cr-indistinguable.

1.5.6 Analysis of failure

Proposition 4. *If two graphs with n vertices with the same features are d-regular[3] then they are cr-indistinguishable.*

Example 5. *Figure 1.3 shows two non isomorphic 3-regular graphs with both 20 vertices. They are not isomorphic because decaprismane has 4-cycles while dodecahedrane does not.*

[3] A graph is d-regular if all vertices have the same degrees d.



(a) Decaprismane. (b) Dodecahedrane.

Figure 1.3: Decaprismane and dodecahedrane. Two non isomorphic graphs that are 1-WL-indistinguishable [Sat20].

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

Theorem 6 ([BES80], [AKRV15])**.** *Let G_n, G'_n be two independent uniformly random graphs with n vertices. We have:*

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

🎓Read [BES80], [AKRV15] and provide a proof of the above theorem

1.6 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 is stated in Theorem VIII.1 in [Gro21], as well as [XHLJ19] [MRF+19].

1.6.1 Colour refinement is self-contained

Theorem 7. *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 8. $\mathit{cr}(G) \sqsubseteq N^{(L)}(G)$.

1.6.2 But GNNs are powerful

Is there a GNN that computes the cr-partition? In Theorem VIII.4 of [Gro21] 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 9. *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*

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

🎓Read the proof of Theorem VIII.4 in [Gro21]

1.7 Generalizations (*)

Figure 1.3 shows two non-isomorphic regular graphs that cr is unable to distinguish. So the natural idea is to generalize 1-WL to tuples of vertices instead of single vertices. We introduce the two main generalizations with the notations of [MLM+23]: k-FWL = k-folkore WL, and k-OWL = k-oblivious WL.

1.7.1 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:

$$\ell'[\vec{u}] := (\ell[\vec{u}], \{\!\{(\ell[\vec{u}_{[1:=w]}], \ldots, \ell[\vec{u}_{[k:=w]}] \mid w \in V)\}\!\})$$

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

1.7.2 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:

$$\ell'[\vec{u}] := (\ell[\vec{u}], (\{\!\{\ell[\vec{u}_{[1:=w]}] \mid w \in V\}\!\}, \ldots, \{\!\{\ell[\vec{u}_{[k:=w]}] \mid w \in V\}\!\}))$$

1.7.3 Relations

Proposition 10. *cr and 2-OWL are as powerful.*

Proposition 11. *k-FWL is as powerful as $(k+1)$-OWL.*

Proposition 12. *$k+1$-OWL is strictly more powerful than k-OWL.*

🎓Prove the propositions

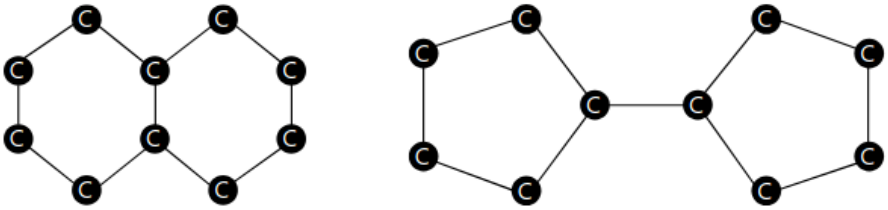
Further reading

1.7.4 Higher-order GNNs

Morris et al. [MRF+19] have proposed generalization GNNs that corresponds to k-OWL for $k > 2$.

Exercises

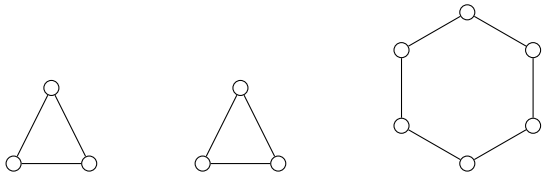
Exercise 2. *Consider these two molecules (from [Sat20]):*



(a) Decalin. (b) Bicyclopentyl.

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

Exercise 3. *Consider the two graphs G and H (Figure 1 in [HV21]):*



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

Exercise 5. *Propose an efficient implementation of algorithm* `cr`.

Chapter 2

... and logic

Key reference: [BKM+20]

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

2.1 First-order logic

First-order logic (FO) validity problem is undecidable [Tur37]. A noticeable decidable fragment is FO_2, the fragment of FO of formulas only containing two variables.

Theorem 13. *[GKV97] The satisfiability problem of FO_2 is NEXPTIME-complete.*

Proof. Upper bound Upper bound is obtained by small model property via Scott's formula. Scott's reduction consists in defining tr such that for all FO_2-formulas φ, $tr(\varphi)$ is in the Gödel class ($\forall\forall\exists^*$-fragment), and φ and $tr(\varphi)$ are equisatisfiable. To do we proceed as follows:

1. First we get rid of predicates of arity > 2 as follows.
   (a) Consider an atomic subformula $R(v_1, \ldots, v_n)$ where $v_1, \ldots, v_n \in \{x, y\}$.
   - If $\{v_1, \ldots, v_n\} = \{x, y\}$, replace $R(v_1, \ldots, v_n)$ by $R^{(v_1,\ldots,v_n)}(x, y)$ where $R^{(v_1,\ldots,v_n)}$ is a fresh binary predicate
   - If $\{v_1, \ldots, v_n\} = \{x\}$, replace $R(x, \ldots, x)$ by $R^{(v_1,\ldots,v_n)}(x)$ where $R^{(x,\ldots,x)}$ is a fresh unary predicate
   - same for y

   (b) We finish by guaranteeing some equivalences. For instance, if $R(x, y, x)$ and $R(y, x, y)$ both appears in φ we add:

   $$\forall x \forall y (R^{(x,y,x)}(x, y) \leftrightarrow R^{(y,x,y)}(y, x)).$$

   etc.

2. Now φ has only predicates of arity at most 2. We now perform a kind of *Tseitin transformation.*
   (a) Each subformula ψ is replaced by a predicate isTrue_ψ of arity 0, 1, 2 depending on the number of free variables in ψ

(b) The final formula is

$$tr(\varphi) := \mathsf{isTrue}_\varphi \wedge \bigwedge_{\psi(\vec{v}) \text{ subformula}} \forall \vec{v}(\mathsf{isTrue}_\psi(\vec{v}) \leftrightarrow meaning_\psi(\vec{v}))$$

where $meaning_\psi$ is:

i. ψ if ψ is atomic;
ii. $\mathsf{isTrue}_\alpha(\vec{v}) \wedge \mathsf{isTrue}_\beta(\vec{v})$ if $\psi = \alpha(\vec{v}) \wedge \beta(\vec{v})$
iii. $\neg\mathsf{isTrue}_\alpha(\vec{v})$ if $\psi = \neg\alpha(\vec{v})$
iv. $\forall v\ \mathsf{isTrue}_\alpha$ if $\psi = \forall v\alpha(\vec{v})$

Note that we get conjuncts with quantifiers $\forall\forall$ for (i-iii). For (iv), because of the $\leftrightarrow$ we get a $\forall\forall$ and a $\forall\exists$.

Now, we can group conjunct and rewrite $tr(\varphi)$ as a formula of the form

$$\forall x\forall y\alpha(x,y) \wedge \bigwedge_{i=1..m} \forall x\exists y\beta_i(x,y)$$

where α and β_i are quantifier-free formulas.

W.l.o.g. we can suppose that $\beta_i(x,y) \models (x \neq y)$. Indeed, if the model $\mathcal{M}$ has at least two elements we have that

$$\mathcal{M} \models (\forall x\exists y\beta_i(x,y) \leftrightarrow \forall x\exists y \underbrace{(x \neq y \wedge (\beta_i(x,x) \vee \beta_i(x,y))}_{\text{the new } \beta_i x,y}).$$

Now we define the notion of type. A *type* $t(\vec{v})$ is MCS over predicates in $tr(\varphi)$ involving only variables in $\vec{v}$. We say 1-type when $\vec{v}$ is variable and 2-type if $\vec{v}$ is (x,y) or (y,x).

Now, we consider a structure $\mathcal{M}$ that satisfies $tr(\varphi)$. We will build a small model from it. Given an element a in the domain of $\mathcal{M}$ the type t_a of a is the unique 1-type satisfied by a.

Same given elements a, b for 2-type.

An element a is a king if a is the only element in the structure to satisfy type t_a. Let K be the set of kings in $\mathcal{M}$.

Let g_i be Skolem function for $\forall x\exists y\beta_i(x,y)$.

$C := K \cup \bigcup_i \{g_i(K)\}$

...

Lower bound

Lower bound can be obtained from the NEXPTIME-hardness, see [Für83] and [GKV97]. We reproduce here the lower bound proof to avoid the reader to navigate throw the different papers. To this aim, we give a reduction from the tiling problem with Wang tiles of a $2^n \times 2^n$-torus where n is given in unary, and $\mathbb{T}$ is the set of tiles, and a give tile seed t_0. We construct a FO_2-formula φ as follows. A variables (eg. x, y) denotes a position of a cell in the torus. We introduce unary predicates $X_i(x)$ for $i = 0..n-1$ that says that the i-th bit of the X-coordinate is 1. Same for $Y_i(y)$ for the Y-coordinate. We introduce also $T_t(x)$ that says that tile t is at x. Before defining φ we introduce the macros:

- $Hsucc(x,y) :=$ according to the $X_i(.)$ and $Y_i(.)$, y is the next cell at right of x
- $Vsucc(x,y) :=$ according to the $X_i(.)$ and $Y_i(.)$, y is the next cell at the top of x

- $eq(x,y) := Heq(x,y) \wedge Veq(x,y)$

Now formula φ is the conjunction of the following formulas:

1. $\forall x$ there is a unique t such that $T_t(x)$
2. same cell is really same cell: $\forall x \bigwedge_{t\in\mathbb{T}}(T_t(x) \rightarrow \forall y, eq(x,y) \rightarrow T_t(y))$
3. $\exists x$ x coordinate is $(0,0)$
4. $\forall x \exists y V succ(x,y)$
5. $\forall x \exists y H succ(x,y)$
6. $\forall x \forall y (H succ(x,y) \rightarrow \bigvee_{t,t' \text{horizontally compatible}} T_t(x) \wedge T_{t'}(y))$
7. same vertically

□

Remark 14. *Five year later, Etessami et al. [EVW02] have studed FO_2 on finite words and ω-words. The corresponding satisfiability problem is also NEXPTIME-complete.*

More generally, FO_k is the fragment of FO of formulas with at most k variables.

2.2 First-order logic with counting

First-order logic with counting (FOC) provides constructions with counting quantification: $\exists^{\geq k} x\varphi$ (there are at least k elements x such that φ holds), $\exists^{\leq k} x$ (there are at most k elements x such that φ holds).

Proposition 15. *FOC and FO (with equality) have the same expressivity.*

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

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

□

However, FOC is interesting because it can lead to interesting fragments such as FOC_2 which is the two-variable fragment of FOC.

Theorem 16. *[Pra14] FOC_2 is NEXPTIME-complete.*

Proposition 17. *[CFI92]We have:*
$\mathsf{cr}(G,u) = \mathsf{cr}(G,v)$ *iff for all* $\varphi(x) \in FOC_2$, $(G,u \models \varphi$ *iff* $G,v \models \varphi)$.

2.3 Modal logic

2.3.1 Syntax

Modal logic 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 6 (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.

2.3.2 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 7 (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 $\neg\Box\neg\varphi$.

2.3.3 Standard translation

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

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

It is interesting to note that ML is a fragment of FO_2.

2.4 Graded Modal logic

2.4.1 Definition

Graded modal logic 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:

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

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.

2.4.2 Link with colour refinement

Theorem 18. *For all rounds t, for all colours c, 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 19. *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 20 ([Gro21], p. 6, Th. V.10)**.** *We have:*

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

Proof. (Proof given in Appendix in [Gro21])

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 colors 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 colors of u and u' after t rounds are different:

$$\begin{aligned} \mathsf{cr}^{(t)}(u) &= (\mathsf{cr}^{(t-1)}(u), \{\!\{.\}\!\}) \\ \neq \mathsf{cr}^{(t)}(u') &= (\mathsf{cr}^{(t-1)}(u'), \{\!\{.\}\!\}) \end{aligned}$$

  Either the first coordinate is different: $\mathsf{cr}^{(t-1)}(u) \neq \mathsf{cr}^{(t-1)}(u')$. So by $\mathcal{P}(r-1)$, there is some formula of modal depth at most r that is true in u and false in u.

  Or the multisets are different. There is a color c such that

$$\mathit{card}(\{v \in E(u) \mid \mathsf{cr}^{(t)}(v) = c\}) \neq \mathit{card}(\{v \in E(u') \mid \mathsf{cr}^{(t)}(v) = c\})$$

  .

  W.l.o.g.

$$\alpha := \mathit{card}(\{v \in E(u) \mid \mathsf{cr}^{(t)}(v) = c\}) > \mathit{card}(\{v \in E(u') \mid \mathsf{cr}^{(t)}(v) = c\})$$

  .

  We set $\varphi := \Diamond^{\geq \alpha}\varphi_{t-1,c}$ where $\varphi_{t-1,c}$ is given by Theorem 18. We have $G, u \models \varphi$ but $G, u \not\models \varphi$.

  Hence not 2.

□

2.4.3 Link with GNNs

Proposition 21 (Prop. 4.1 [BKM+20]). *For all GML-formula φ, there is a GNN N such that for all G, u we have:*

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

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

Proposition 22 (Prop. 4.2 [BKM+20]). *For all GNN N that is FO-expressible, then there is a GML-formula φ such that*

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

It is sad to restrict ourselves to GNN that ar FO-expressible.

Exercises

Exercise 6. *Prove that there is 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 7. *Prove that ML has the expressivity than GNNs where the aggregation function is MAX instead of SUM. See* `https://arxiv.org/abs/2507.18145`

Exercise 8. *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 L, the unvarallings up to L of G, u and G', u' are isomorphic.

Exercise 9. *In this exercise, we will prove that any GNN that is FO-definable is captured by a GML-formula.*

1. *Show that if for all L, the unravellings up to 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 3

Verifying GNNs

Key reference: [NSST24] and [BLMT24]

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.

3.1 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. The language is inspired from the one in [SST25].

3.1.1 Syntax

We consider expressions generated by the following grammar:

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

where c is any number, x_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).

3.1.2 Semantics

The semantics mimics the computation performed by a GNN:

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

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

3.1.3 Correspondence with GNNs

Proposition 23. *Given a GNN N, there exists an expression ϑ such that $[\![N]\!] = [\![\vartheta \geq 1]\!]$.*

See [SST25] for a formal proof. We explain this via example. Consider a two-layer GNN $\mathcal{A}$ with input and output dimension 2, using summation for aggregation, activation via $\alpha(x) := \max(0, \min(1, x))$—the truncated ReLU—and a classification function $2x_1 - x_2 \geq 1$. The combination functions are:

$$\begin{aligned}
comb_1((x_1, x_2), (y_1, y_2)) &:= \begin{pmatrix} \sigma(2x_1 + x_2 + 5y_1 - 3y_2 + 1) \\ \sigma(-x_1 + 4x_2 + 2y_1 + 6y_2 - 2) \end{pmatrix}, \\
comb_2((x_1, x_2), (y_1, y_2)) &:= \begin{pmatrix} \sigma(3x_1 - y_1) \\ \sigma(-2x_1 + 5y_2) \end{pmatrix}.
\end{aligned}$$

Then, the corresponding GNN-logic formula $\varphi_{\mathcal{A}}$ is given by:

$$\begin{aligned}
\psi_1 &= \alpha(2x_1 + x_2 + 5agg(x_1) - 3agg(x_2) + 1), \\
\psi_2 &:= \alpha(-x_1 + 4x_2 + 2agg(x_1) + 6agg(x_2) - 2), \\
\chi_1 &:= \alpha(3\psi_1 - agg(\psi_1), \\
\chi_2 &:= \alpha(-2\psi_1 + 5(agg(\psi_2)), \\
\varphi_A &:= 2(\chi_1) - \chi_2 \geq 1.
\end{aligned}$$

Remark 24. *Note that some expressions do not represent a GNN. For instance, $agg(2 \times x)$ does not syntactically correspond to a GNN. Indeed, aggregation is always computed on values of the form $\alpha(...)$.*

3.2 Logic $K^{\#}$

We now define logic $K^{\#}$ defined in [NS23] and [NSST24]. A similar logic is defined in [BLMT24] but it does not have the 1_φ construction. It has the same expressivity but probably not the same succinctness.

3.2.1 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:

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

3.2.2 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 25. *Consider the pointed graph G, u shown in Figure 3.1. We have $G, u \models p \wedge (\#\neg p \geq 2) \wedge \#(\#p \geq 1) \leq 1$. 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.*

3.3 $K^{\#}$ and truncReLU-GNNs

In this section, we consider GNNs where the activation function α is truncReLU.

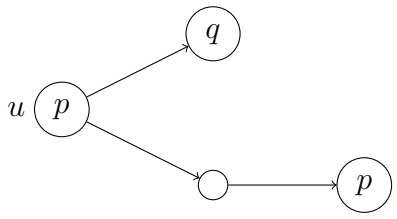


Figure 3.1: Example of a pointed graph G, u. We indicate true propositional variables at each vertex.

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

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

Note that if constants c are integers, the weights in the produced GNN $tr(\varphi)$ are integers.

Proposition 26. *For all $K^{\#}$-formulas φ, $[\![tr(\varphi)]\!]_{G,u} \in \{0,1\}$.*

Proof. By definition of tr. □

Proposition 27. *For all $K^{\#}$-formulas φ, $[\![\varphi]\!] = [\![tr(\varphi) \geq 1]\!]$.*

Proof. We prove it by mutual induction on φ and on ϑ that:

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

- $G, u \models x_i = 1$ iff $[\![x_i]\!]_{G,u} = 1$;
-

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

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

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

□

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

We first suppose that weights are integers. The following translation function tr' takes a GNN described as an expression ϑ and gives an equivalent $K^{\#}$-expression.

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

Proposition 28. *For all GNNL-expressions ϑ, $[\![\vartheta \geq 1]\!] = [\![tr'(\vartheta) \geq 1]\!]$.*

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

Translating GNNs with rational weights. So far we handled GNNs with integer weights. In order to handle weights that are rationals, we multiply all numbers by a constant M so that we get integers.

The translation is then:

$$\begin{aligned} tr'(x_i) &= M \times x_i \\ tr'(c) &= M \times c \\ tr'(c\vartheta) &= M \times c \times tr'(\vartheta) \\ tr'(\vartheta + \vartheta') &= tr'(\vartheta) + tr'(\vartheta') \\ tr'(\mathsf{truncReLU}(\vartheta)) &= 1_{tr'(\vartheta)=1} + 2 \times 1_{tr'(\vartheta)=2} + \cdots + M \times 1_{tr'(\vartheta)=M} \\ tr'(agg(\vartheta)) &= \#(tr'(\vartheta) = 1) + 2 \times \#(tr'(\vartheta) = 2) + \ldots M \times \#(tr'(\vartheta) = M) \\ & \qquad \text{provided } \vartheta \text{ is of the form } \mathsf{truncReLU}(.) \end{aligned}$$

In the above translation, the activation function $\mathsf{truncReLU}$ is simulated by $\mathsf{truncReLU}_M$ defined by:

$$\mathsf{truncReLU}_M(x) := max(0, min(M, x)).$$

Proposition 29. *For all GNNL-expressions ϑ, $[\![\vartheta \geq 1]\!] = [\![tr'(\vartheta) \geq M]\!]$.*

In [BLMT24], the authors explain a methodology to capture any activation functions that is piece-wise linear and eventually constant.

3.4 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 28 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.

3.5 Going further

3.5.1 ReLU

Handling ReLU is more involved, and we do not have a clear correspondence with modal logic yet [BLMT24]. In [HZ19], they generalize existential Presburger arithmetics with Kleene star as follows. The Kleene star of a set $M \subseteq \mathbb{Z}^d$ of vectors is defined by:

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

The obtained logic is *Existential Presburger arithmetic with star* $\exists PA^*$. 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:

$$[\![\vec{a} \cdot \vec{z} \geq c]\!] = \{\vec{x} \in \mathbb{Z}^n \mid \vec{a} \cdot \vec{x} \geq c\}$$
$$[\![\varphi^*]\!] = [\![\varphi]\!]^*$$

Theorem 30. *(Th. III.1 in [HZ19]) The satisfiability problem of $\exists PA^*$ is NEXPTIME-complete, and NP-complete if the number of nested star is bounded.*

Corollary 31. *(Th. 6.20 in [BLMT24]) The satisfiability problem of ReLU-GNNs is NEXPTIME-complete, and NP-complete if the number of layers is bounded.*

Proof.
- Upper bound. (Th. 6.26 in [BLMT24]). We reduce to the satisfiability of $\exists PA^*$. The Kleene-star is used to perform the aggregation over an unbounded number of successors.

- Lower bound. (Th. 6.28 in [BLMT24]) We reduce the $\exists PA^*$-satisfiability to the satisfiability of a GNN with ReLU.

□

Our GNN-logic and $\exists PA^*$ are very similar, but they also differ. The semantics of a qL-expression is a single real number, whereas the semantics of a $\exists PA^*$ 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 $\exists PA^*$ the structure is implicit. If we have several formulas $(..)*$, then the successors are different. That is why in [BLMT24] 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.

3.5.2 Quantized GNNs

In [SST25], the authors prove that the satisfiability problem of GNN-logic when numbers are quantized (e.g. 32-bit arithmetics) is in PSPACE-complete. The bitwidth n is the number of bits used to represent the numbers.

Theorem 32. *Verifying quantized GNNs where the bitwidth n is given in unary is PSPACE-complete.*

Open questions

- Parametrized complexity of $\exists PA^*$ 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
- Tableau method of a logic for GNNs with ReLU

Exercises

Exercise 10. *Write a translation from GML into $K^{\#}$ preserving the semantics.*

Exercise 11. *Prove that $K^{\#}$ is more expressive than FO.*

Hint: see Appendix in [NSST24]

Chapter 4

Satisfiability problem of K

Before tackling the satisfiability problem for $K^{\#}$, 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.

4.1 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 8 (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:

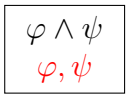
$\neg\Box\varphi$	becomes	$\Diamond\neg\varphi$
$\neg\Diamond\varphi$	becomes	$\Box\neg\varphi$
$\neg(\varphi \wedge \psi)$	becomes	$(\neg\varphi \vee \neg\psi)$
$\neg(\varphi \vee \psi)$	becomes	$(\neg\varphi \wedge \neg\psi)$

4.2 Overview

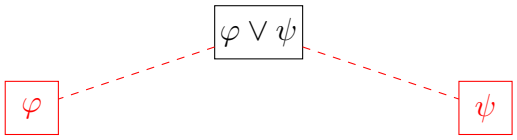
In a nutshell, the tableau method is a proof system that constructs a Kripke structure. Each time a formula is written, it means that the formula *should* hold at a given world. It can be seen as a procedure that rewrites a labelled graph. The tableau method starts with an initial graph made up of one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

4.3 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).



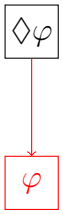
The following rule for the $\vee$ connective is non-deterministic.



The clash rule for contradiction gives a clash.

clash

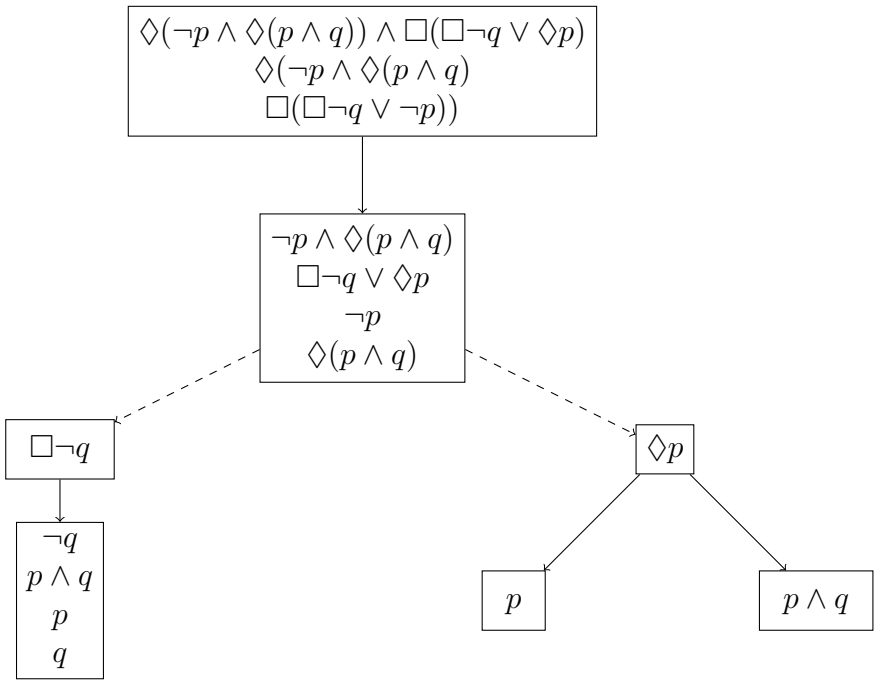
The rule for $\Diamond$ adds a successor containing φ if there is no successor.



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

4.4 Example

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



4.5 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. Let us start by defining the rules for Boolean connectives:

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

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

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.

4.6 Soundness and completeness

Theorem 33 (soundness)**.** *If there is an execution of the tableau method that is not clashing, then the initial formula is satisfiable.*

Theorem 34 (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 [BdRV01] (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 [Tob99].

Concerning implementation issues and optimisation, the reader may have a look to [HHSS07].

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 5

Satisfiability problem of K 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 4 we just saw into a non-deterministic algorithm.

5.1 Algorithm

The design of our algorithm is as follows.

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

```
function satK(Γ)
    choose outcomes of Boolean rules until Γ is saturated
    if the clash rule can be applied on Γ then
        reject
    for all formulas of the form ◇ψ in Γ
        satK(ψ ∪ {χ | □χ ∈ Γ})
    accept
```

5.2 Soundness and completeness

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

Theorem 35 (completeness)**.** *If* Γ *is satisfiable, then there exists an accepting 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 an accepting 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 accepting.

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

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 accepting execution. We can thus construct an accepting execution of satK(Γ). □

Theorem 36 (soundness)**.** *If sat*(Γ) *has an accepting 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*(Γ) has an accepting 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 $P(k+1)$.

Let us take Γ of model depth k+1 such that *sat*(Γ) 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 5.1, 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 $\mathcal{M}, w \models \varphi$.

□

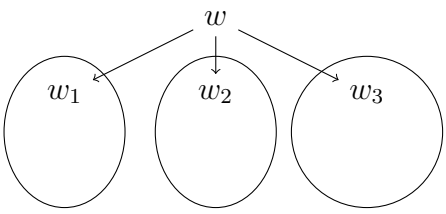


Figure 5.1: Model constructed by gluing models obtained from the subcall when diamond formulas are $\Diamond\psi_1, \Diamond\psi_2$ and $\Diamond\psi_3$.

5.3 PSPACE upper bound

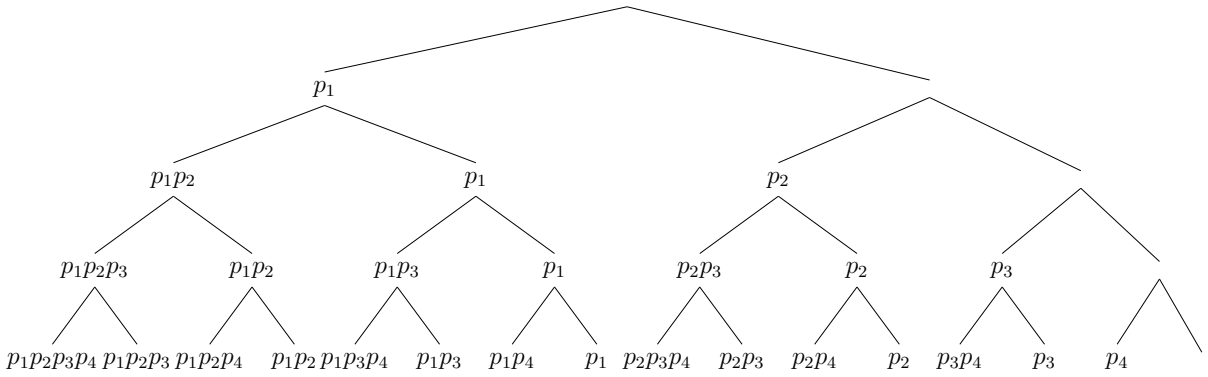
Theorem 37. *The satisfiability problem for K is in PSPACE.*

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. □

5.4 PSPACE lower bound

Theorem 38. *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.



Modal logic can express that the Kripke model contains this tree. The following formula explains how the i-th level should look like:

$$\Box^i[\Diamond p_i \wedge \Diamond\neg p_i \wedge \bigwedge_{j<i}(p_j \leftrightarrow \Box p_j)]$$

Let $TREE$ be the conjunction for $i = 1..2n$ that forces the Kripke model to contain the binary tree up to level $2n$. On the input $\exists p_1 \forall p_2 \ldots \exists p_{2n-1} \forall p_{2n} \chi$, the reduction computes in polynomial time the modal formula $TREE \wedge \Diamond\Box ... \Diamond\Box\chi$. The former is true iff the latter is satisfiable.

□

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 [CR02]

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 6

Satisfiability problem of logic with counting

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

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

As explained in [NS23], 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:

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

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).

6.2 Quantifier-free Fragment Boolean Algebra with Presburger Arithmetic

A QFBAPA formula is propositional formula where each atom is either an inclusion of sets or equality of sets or linear constraints [KR07]. 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 $|S|$ denoting the cardinality of the set denoted by the set expression S. Let us give a formal definition of the syntax and semantics.

Definition 9. *(see Figure 1 in [KR07]) A QBFBAPA formula is generated by the axiom φ in the following BNF grammar:*

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

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 original QFBAPA [KR07] 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 10. *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 and set expressions 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 11. *The truth conditions are given as follows:*

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

We are going to prove:

Theorem 39. *QFBAPA satisfiability problem is in NP.*

6.3 Fail for proving NP by naïve argument

We first discuss the fact that knowing the size (written in binary) does not help much. Consider the following formula.

$$|\mathcal{U}| = n \ \wedge \bigwedge_{0 \leq i < j \leq m} |S_i \cup S_j| = 30 \ \wedge \bigwedge_{0 \leq i \leq m} |S_i| = 20$$

A certificate would consist in telling for each set variable S_i which elements are in $[\![S_i]\!]$. So each set variable is represented by a word $\{0,1\}^n$ while n is given in binary. The certificate is of exponential size in the size of the formula. So it seems that the satisfiability problem of QFBAPA is in NEXPTIME.

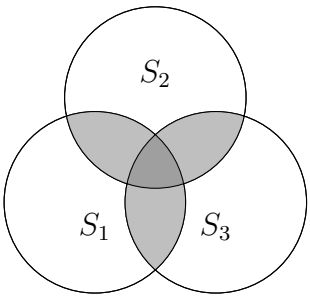


Figure 6.1: A Venn diagram with 3 set variables is made up of 8 regions.

6.4 Venn diagrams

A Venn diagram is a picture that contains all the possible intersections called *regions*, see Figure 6.1.

The idea is to reason about the size of each region obtained by intersection and introducing a variable $\ell_{01010110}$ to denote that size:

Regions	Size of that region
$\overline{S_1} \cap \overline{S_2} \cap \cdots \cap \overline{S_n}$	$\ell_{000\ldots0}$
$\overline{S_1} \cap \overline{S_2} \cap \cdots \cap S_n$	$\ell_{000\ldots1}$
$\vdots$	$\vdots$
$S_1 \cap S_2 \cap \cdots \cap \overline{S_n}$	$\ell_{111\ldots0}$
$S_1 \cap S_2 \cap \cdots \cap S_n$	$\ell_{111\ldots1}$

Example 40. *For instance for $n = 3$ we have:*

Regions	*Size of that region*
$\overline{S_1} \cap \overline{S_2} \cap \overline{S_3}$	ℓ_{000}
$\overline{S_1} \cap \overline{S_2} \cap S_3$	ℓ_{001}
$\overline{S_1} \cap S_2 \cap \overline{S_3}$	ℓ_{010}
$\overline{S_1} \cap S_2 \cap S_3$	ℓ_{011}
$S_1 \cap \overline{S_2} \cap \overline{S_3}$	ℓ_{100}
$S_1 \cap \overline{S_2} \cap S_3$	ℓ_{101}
$S_1 \cap S_2 \cap \overline{S_3}$	ℓ_{110}
$S_1 \cap S_2 \cap S_3$	ℓ_{111}

We could rewrite any QFBAPA-formula using variables $\ell_{0101010111}$ as follows. We replace each cardinality expression with sums of the appropriate variables $\ell_{0101010111}$. For instance:

$$|S_1 \cap S_2 \cap \overline{S_3}| = \ell_{110}$$
$$|S_1| = \ell_{100} + \ell_{101} + \ell_{110} + \ell_{111}$$

But there are an exponential number of variables in the number of set variables that will be used. So again: the satisfiability problem of QFBAPA is in NEXPTIME.

Later on we will see that a poly-number of non-empty regions is sufficient. From that, we get NP membership.

6.5 Naïve reduction to QFPA

We first explain the naïve reduction to quantifier-free Presburger arithmetics. It is made of several steps of rewriting.

Getting rid of inclusions and set equality. We rewrite $b_1 = b_2$ into $b_1 \subseteq b_2 \wedge b_2 \subseteq b_1$. We rewrite $b_1 \subseteq b_2$ into $|b_1 \cap \overline{b_2}| = 0$.

Variables for cardinalities of Boolean expressions. Instead of writing $|b_i|$, we introduce a new integer variable k_i that represent the cardinal of b_i. Our QFBAPA-formula φ is written into

$$\underbrace{\varphi[|b_i| := k_i]}_{\text{PA formula}} \wedge \underbrace{\bigwedge_{i=1}^{d} |b_i| = k_i}_{\psi}.$$

Said otherwise, we replace $|b_i|$ by k_i and enforce the equalities $|b_i| = k_i$ in a separate clause ψ.

W.l.o.g. we suppose that $b_1 = \mathcal{U}$.

Venn diagrams. Now, the main idea is to rewrite ψ. The goal is to get rid from $\cap$, $\cup$, etc. and only have integer variables. To do that, we will introduce integer variables $\ell_{1010010}$ etc. to represent cardinals of regions. At the end ψ will be replaced by ψ' which is a formula free from Boolean algebra operators ($\cap$, $\cup$, etc. are deleted).

Let $S_1, \ldots, S_e$ the set variables appearing in $b_1, \ldots, b_d$.

Each Venn diagram region is represent by a string $\beta \in \{0,1\}^e$. We write, given $taken \in \{0,1\}$:

$$S_i^{taken} = \begin{cases} S_i \text{ if } taken = 1 \\ \overline{S_i} \text{ if } taken = 0. \end{cases}$$

The Venn diagram region corresponding to β is:

$$R_\beta := \bigcap_{j=1}^{e} S_i^{\beta_i}.$$

Example 41. $R_{101001} = S_1 \cap \overline{S_2} \cap S_3 \cap \overline{S_4} \cap \overline{S_5} \cap S_6$.

Given an set expression b we can say whether a given s_β is included in b.

Example 42. R_{101001} *is included in* $S_1 \cap S_3$.

To check that s_β is included in b, we can consider b as a propositional formula and β as a valuation. If β satisfies b then s_β is included in b. In the sequel, we write $\beta \models b$.

Example 43. $\beta = 101001$ *is the valuation*

$$\left\{\begin{array}{l} S_1 := 1 \\ S_2 = 0 \\ S_3 = 1 \\ S_4 = 0 \\ S_5 = 0 \\ S_6 = 1 \end{array}\right\}$$

and it satisfies the formula $S_1 \wedge S_3$.

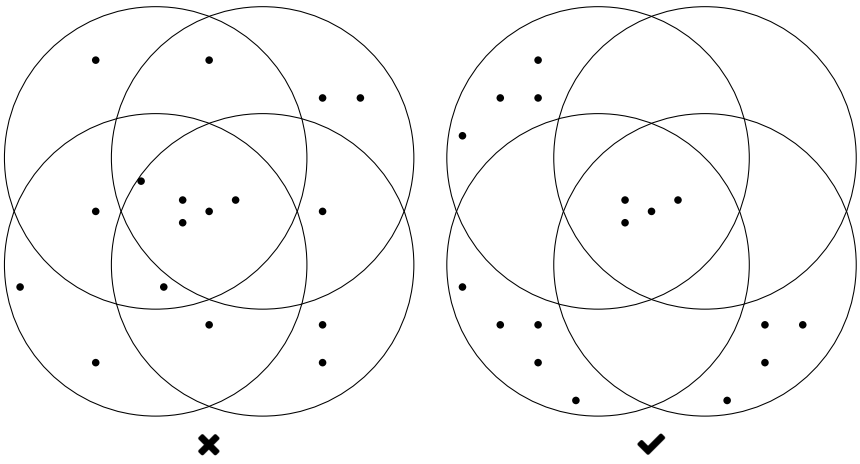
We introduce variable ℓ_β to represent $|R_\beta|$. We rewrite ψ into ψ':

$$\psi' := \bigwedge_{i=1}^{d} \sum_{\beta \in \{0,1\}^e | \beta \models b_i} \ell_\beta = k_i.$$

Proposition 44. *φ is QFBAPA-satisfiable iff $\varphi[|b_i| := k_i] \wedge \psi'$ is QFPA-satisfiable.*

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

We will show that only a polynomial number of ℓ_β can be non-zero.



To do that, we will apply a Carathéodory bound for integer cones, see Th. 1 (ii) in [ES06], reformulated by the following lemma. 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, 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.

Lemma 45. *[ES06] 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). That is $M < \frac{2^{|X|/2d}}{4d}$.

$$
\begin{aligned}
d\log(2|X|M+1) &< d\log\left(\frac{|X|}{2d}2^{|X|/(2d)}+1\right) && \text{by assumption}\\
&\leq d\log\left(2^{|X|/(2d)}\left(\frac{|X|}{2d}+1\right)\right)\\
&= \frac{|X|}{2}+d\log\left(\frac{|X|}{2d}+1\right)\\
&\leq \frac{|X|}{2}+d\cdot\frac{|X|}{2d} \text{ by concavity of log}\\
&= |X|.
\end{aligned}
$$

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, otherwise we are done).

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

$$
\begin{aligned}
card(\left\{\sum_{x\in\tilde{X}}x \mid \tilde{X}\subseteq X\right\}) &\leq (2|X|M+1)^d\\
&< 2^{|X|}.
\end{aligned}
$$

So 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:

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

We have

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

W.l.o.g. we suppose that $A'\neq\emptyset$. We set $\lambda:=\min_{x\in A'}\lambda_x$. Then:

$$
\begin{aligned}
b=\sum_{x\in X}\lambda_x x &= \sum_{x\in X\setminus A'}\lambda_x x+\sum_{x\in A'}\lambda_x x\\
&= \sum_{x\in X\setminus A'}\lambda_x x+\sum_{x\in A'}(\lambda_x-\lambda)x+\lambda\sum_{x\in A'}x\\
&= \sum_{x\in X\setminus A'}\lambda_x x+\sum_{x\in A'}(\lambda_x-\lambda)x+\lambda\sum_{x\in B'}x\\
&= \sum_{x\in A'}(\lambda_x-\lambda)x+\sum_{x\in X\setminus(A'\cup B')}\lambda_x x+\sum_{x\in B'}(\lambda_x+\lambda)
\end{aligned}
$$

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 $|X| \leq 2d \log_2(4dM)$.

□

To apply Lemma 45, we rewrite ψ' as the following system with d equations:

$$\begin{cases} \sum_{\beta \in \{0,1\}^e} \ell_\beta [\![b_1]\!]_\beta = k_1 \\ \vdots \\ \sum_{\beta \in \{0,1\}^e} \ell_\beta [\![b_d]\!]_\beta = k_d \end{cases}$$

where

$$[\![b_i]\!]_\beta = \begin{cases} 1 \text{ if } \beta \models b_i \\ 0 \text{ otherwise} \end{cases}.$$

In a vectorial form, we get:

$$\sum_\beta \ell_\beta \begin{pmatrix} [\![b_1]\!]_\beta \\ \vdots \\ [\![b_d]\!]_\beta \end{pmatrix} = \begin{pmatrix} k_1 \\ \vdots \\ k_d \end{pmatrix}$$

Said differently, if we set $X = \left\{ \begin{pmatrix} [\![b_1]\!]_\beta \\ \vdots \\ [\![b_d]\!]_\beta \end{pmatrix} \mid \beta \in \{0,1\}^e \right\}$, we get:

$$\begin{pmatrix} k_1 \\ \vdots \\ k_d \end{pmatrix} \in \texttt{cone}(X).$$

By Lemma 45, there exists a subset $\mathbb{B} \subseteq \{0,1\}^e$ of size at most $2d \log_2(4d)$ such that

$$\sum_{\beta \in \mathbb{B}} \ell_\beta \begin{pmatrix} [\![b_1]\!]_\beta \\ \vdots \\ [\![b_d]\!]_\beta \end{pmatrix} = \begin{pmatrix} k_1 \\ \vdots \\ k_d \end{pmatrix}.$$

6.7 QBFPAPA satisfiability in NP

Here is an algorithm for testing the satisfiability problem of QFBAPA-formula φ.

input: a QFBAPA-formula φ
output: true iff φ is QFBAPA-satisfiable
function QFBAPAsat(φ)
    d := number of Boolean set expressions
    e := number of set variables
    Guess a subset $\mathbb{B} \subseteq \{0,1\}^e$ of size $2d \log_2(4d)$
    Check whether the QFPAPA-formula $\varphi[|b_i| := k_i] \wedge \bigwedge_{i=1}^d \sum_{\beta \in \mathbb{B} | \beta \models b_i} \ell_\beta = k_i$ is satisfiable
    If yes, **return** true. Otherwise **return** false

Example 46. *Consider the formula* $(|S \cap T| \leq 5) \wedge (|S| > |T|)$. *We have* $e = 2$ *set variables:* S *and* T. *We have* $d = 3$ *set expressions:* $S \cap T$, S *and* T.

Theorem 47. *QFBAPA satisfiability problem is in NP.*

Proof. We have to prove that QFBABAsat is sound and complete. We have φ QFBAPA-satisfable iff $\varphi[|b_i := k_i] \wedge \bigwedge_{i=1}^{d} \sum_{\beta \in \{0,1\}^e | \beta \models b_i} \ell_\beta = k_i$ is QFBAPA-satisfiable.

$\boxed{\Rightarrow}$ If φ has a model, then $\varphi[|b_i := k_i] \wedge \bigwedge_{i=1}^{d} \sum_{\beta \in \{0,1\}^e | \beta \models b_i} \ell_\beta = k_i$ has a model: interpret k_i as the cardinality of b_i and ℓ_β as the cardinality of the region R_β.

$\boxed{\Leftarrow}$ If $\varphi[|b_i := k_i] \wedge \bigwedge_{i=1}^{d} \sum_{\beta \in \{0,1\}^e | \beta \models b_i} \ell_\beta = k_i$ is satisfiable, construct regions R_β with ℓ_β each. Interpret b_i as the union of R_β such that $\beta \models b_i$.

Finally, QFBAPAsat is non-deterministic algorithm that runs in polynomial time in $|\varphi|$. □

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

We write a non-deterministic procedure inspired from [NS23] (in which we forgot to use QFBAPA) and [Baa17] (in which QFBAPA is used but for a description logic close to $K^\#$).

6.8.1 Description of the algorithm

input: Γ a set of $K^\#$-formulas
output: yes if Γ is $K^\#$-satisfiable
function $\mathsf{sat}K^\#(\Gamma)$
  apply non-deterministically Boolean tableau rules to Γ
  let S be the set of inequalities in Γ
  let $\#\psi_1, \ldots \#\psi_d$ be a list of all constructions of the form $\#\psi$ appearing in Γ
  Guess $\mathbb{B} \subseteq \{0,1\}^d$ of size $\leq 2d \log_2(4d)$
  Replace in S each occurrence of $\#\psi_i$ by $\sum_{\beta \in \mathbb{B} | \beta_i = 1} \ell_\beta$
  Add the constraint $\ell_\beta \geq 1$ for all $\beta \in \mathbb{B}$ in S
  Check that S is QFBAPA-satisfiable
  for $\beta \in \mathbb{B}$ **do**
    $\mathsf{sat}K^\#(\bigwedge_{i|\beta_i=0} \neg\psi_i \wedge \bigwedge_{i|\beta_i=1} \psi_i)$

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 directly handled by tableau rules. 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 (ℓ_{100} is the cardinal);
2. $\psi_1 \wedge \neg\psi_2 \wedge \psi_3$ holds in a subset of successors (ℓ_{101} is the cardinal);
3. $\psi_1 \wedge \psi_2 \wedge \psi_3$ holds in a subset of successors (ℓ_{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 ℓ_{100} (if 100 in $\mathbb{B}$), ℓ_{101} (if 101 in $\mathbb{B}$), ℓ_{110} (if 110 in $\mathbb{B}$), ℓ_{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 $\ell_\beta \geq 1$ for all non-zero regions β.

6.8.2 Soundness and completeness

Proposition 48. *If* $\mathsf{sat}K^{\#}(\Gamma)$ *has an accepting execution then* Γ *is* $K^{\#}$*-satisfiable.*

Proof. We glue together the set of successors etc. We obtain a tree whose root satisfy Γ. □

Proposition 49. *If* Γ *is* $K^{\#}$*-satisfiable, then* $\mathsf{sat}K^{\#}(\Gamma)$ *has an accepting execution.*

Proof. We apply the tableau rules accordingly. □

Exercises

Exercise 12. *Take a* $K^{\#}$*-formula of your choice and apply the algorithm* $\mathsf{sat}K^{\#}$*.*

Exercise 13. *Show that QFBAPA is NP-hard even if numbers that are written are in formulas are 0 and 1.*

Exercise 14. *Prove formally the theorems of the chapter.*

Exercise 15. *Adapt the algorithm* $\mathsf{sat}K^{\#}$ *when we are search for an undirected graph.*

Chapter 7

Going further

This chapter gives pointers to articles that students may study and present.

7.1 Verification

Exact Verification of Graph Neural Networks with Incremental Constraint Solving: `https://arxiv.org/pdf/2508.09320`

Fundamental Limits in Formal Verification of Message-Passing Neural Networks: `https://openreview.net/forum?id=WlbG820mRH-`

7.2 Expressivity

Aggregate-Combine-Readout GNNs Are More Expressive Than Logic $C2$: `https://arxiv.org/pdf/2508.06091`

7.3 QFBAPA

Lower bounds of Caratheodory for QBFPABA in [KR07].

Bibliography

[AKRV15] Vikraman Arvind, Johannes Köbler, Gaurav Rattan, and Oleg Verbitsky. On the power of color refinement. In Adrian Kosowski and Igor Walukiewicz, 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, 2015.

[Baa17] Franz Baader. A new description logic with set constraints and cardinality constraints on role successors. In Clare Dixon and Marcelo Finger, 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, 2017.

[Bab16] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In Daniel Wichs and Yishay Mansour, 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, 2016.

[BBG17] Christoph Berkholz, Paul S. Bonsma, and Martin Grohe. Tight lower and upper bounds for the complexity of canonical colour refinement. *Theory Comput. Syst.*, 60(4):581–614, 2017.

[BdRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[BES80] László Babai, Paul Erdös, and Stanley M. Selkow. Random graph isomorphism. *SIAM J. Comput.*, 9(3):628–635, 1980.

[BKM+20] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. 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, 2020.

[BLMT24] Michael Benedikt, Chia-Hsuan Lu, Boris Motik, and Tony Tan. Decidability of graph neural networks via logical characterizations. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, 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, 2024.

[CC82] A. Cardon and Maxime Crochemore. Partitioning a graph in o(|a| log2 |v|). *Theor. Comput. Sci.*, 19:85–98, 1982.

[CFI92] Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Comb.*, 12(4):389–410, 1992.

[CR02] Alexander V. Chagrov and Mikhail N. Rybakov. How many variables does one need to prove pspace-hardness of modal logics. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, 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, 2002.

[ES06] Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Oper. Res. Lett.*, 34(5):564–568, 2006.

[EVW02] Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2):279–295, 2002.

[Für83] Martin Fürer. The computational complexity of the unconstrained limited domino problem (with implications for logical decision problems). In Egon Börger, Gisbert Hasenjaeger, and Dieter Rödding, editors, *Logic and Machines: Decision Problems and Complexity, Proceedings of the Symposium "Rekursive Kombinatorik" held from May 23-28, 1983 at the Institut für Mathematische Logik und Grundlagenforschung der Universität Münster/Westfalen*, volume 171 of *Lecture Notes in Computer Science*, pages 312–319. Springer, 1983.

[GKV97] Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bull. Symb. Log.*, 3(1):53–69, 1997.

[Gro21] Martin Grohe. 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, 2021.

[HHSS07] Ian Horrocks, Ullrich Hustadt, Ulrike Sattler, and Renate A. Schmidt. Computational modal logic. In Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 181–245. North-Holland, 2007.

[HV21] Ningyuan Teresa Huang and Soledad Villar. 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, 2021.

[HZ19] Christoph Haase and Georg Zetzsche. 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, 2019.

[IL90] Neil Immerman and Eric Lander. 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, 1990.

[KR07] Viktor Kuncak and Martin Rinard. Towards efficient satisfiability checking for boolean algebra with presburger arithmetic. In Frank Pfenning, editor, *Automated Deduction – CADE-21*, pages 215–230, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[MLM+23] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, and Karsten M. Borgwardt. Weisfeiler and leman go machine learning: The story so far. *J. Mach. Learn. Res.*, 24:333:1–333:59, 2023.

[MRF+19] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 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, 2019.

[NS23] Pierre Nunn and François Schwarzentruber. A modal logic for explaining some graph neural networks. *CoRR*, abs/2307.05150, 2023.

[NSST24] Pierre Nunn, Marco Sälzer, François Schwarzentruber, and Nicolas Troquard. 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, 2024.

[Pra14] Ian Pratt-Hartmann. Logics with counting and equivalence. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 76:1–76:10. ACM, 2014.

[PT87] Robert Paige and Robert Endre Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987.

[RNE+22] Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, and Pascal Friederich. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(93), 2022.

[Sat20] Ryoma Sato. A survey on the expressive power of graph neural networks. *CoRR*, abs/2003.04078, 2020.

[Sch88] Uwe Schöning. Graph isomorphism is in the low hierarchy. *J. Comput. Syst. Sci.*, 37(3):312–323, 1988.

[SLJ21] Amirreza Salamat, Xiao Luo, and Ali Jafari. Heterographrec: A heterogeneous graph-based neural networks for social recommendations. *Knowl. Based Syst.*, 217:106817, 2021.

[SST25] Marco Sälzer, François Schwarzentruber, and Nicolas Troquard. Verifying quantized graph neural networks is pspace-complete. *CoRR*, abs/2502.16244, 2025.

[Tob99] Stephan Tobies. A pspace algorithm for graded modal logic. In Harald Ganzinger, 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, 1999.

[Tur37] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, s2-42(1):230–265, 1937.

[XHLJ19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 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, 2019.

[XXC+21] Jiacheng Xiong, Zhaoping Xiong, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6):1382–1393, 2021.

[YKS+22] Zi Ye, Yogan Jaya Kumar, Goh Ong Sing, Fengyan Song, and Junsong Wang. A comprehensive survey of graph neural networks for knowledge graphs. *IEEE Access*, 10:75729–75741, 2022.