Lecture 4: Real computation and arithmetic circuits

Our focus so far

- Indistinguishability of GNNs:
  - When two input graphs can be separated by a GNN?
  - Input graphs can have numerical features in $\mathbb{R}^p$ or abstract features.
- Expressivity characterisations:
  - What graph/node properties can be expressed with GNNs?
  - Input: A graph with features in some finite set. A graph embedding is used to give features in $\mathbb{R}^p$.
  - Output: A classification of (pointed) graphs to Boolean (or finitely many) classes.
  - Type of characterisation: $M_s(G, n) = 1$ if and only if $G, n \models \varphi_s$.

Part 7: GNNs and arithmetic circuits

- What do we cover next?
  - Consider GNNs as devices computing real labelled graph transformations.
    - Meaning, functions of type $\mathcal{G}[\mathbb{R}^{P_1}] \to \mathcal{G}[\mathbb{R}^{P_2}]$.
    - We omit the embedding part and classification part.
    - This gives a cleaner picture of the computation power of GNNs.
    - We relate their power to a mature field of study; circuit complexity theory.
  - Short introduction to arithmetic circuits and circuit complexity classes.
  - Circuit Graph Neural Networks $\approx$ arithmetic circuit families
  - Recursive Circuit Graph Neural Networks $\approx$ recursive arithmetic circuit families
- Main sources:
  - Timon Barlag, Vivian Holzapfel, Laura Strieker, Jonni Virtema and Heribert Vollmer. Graph Neural Networks and Arithmetic Circuits. NeurIPS 2024.
  - Timon Barlag, Vivian Holzapfel, Laura Strieker, Jonni Virtema and Heribert Vollmer. Recurrent Graph Neural Networks and Arithmetic Circuits. KR 2026 (in two weeks).
  - Many things below from slides by Laura and Vivian.

What GNNs compute?

- Recall that a GNN-layer updates feature vectors via

$$g'(n) \coloneqq \mathrm{Comb}\Big(g(n), \mathrm{Agg}\big(\{\{g(m) \mid m \in E(n)\}\}\big)\Big).$$

  where Comb and Agg are simple (often piecewise linear) functions.
- We stipulate that Comb and Agg can be written using $+$, $\times$, and some activation functions σ_i.
- Given graph (G, v) and an l-layer GNN, we can write an expression using $+$, $\times$, σ that computes the final feature vector of v after l-layers.
- In this level of abstraction, we can relate GNNs with arithmetic circuit complexity classes! If we stipulate that Comb and Agg come from circuit complexity classes!

What GNNs compute?

- Recall that a GNN-layer updates feature vectors via

$$g'(n) \coloneqq \mathrm{Comb}\Big(g(n), \mathrm{Agg}\big(\{\{g(m) \mid m \in E(n)\}\}\big)\Big).$$

  where Comb and Agg are simple (often piecewise linear) functions.
- We stipulate that Comb and Agg can be written using $+$, $\times$, and some activation functions σ_i.
- Given graph (G, v) and an l-layer GNN, we can write an expression using $+$, $\times$, σ that computes the final feature vector of v after l-layers.
- In this level of abstraction, we can relate GNNs with arithmetic circuit complexity classes! If we stipulate that Comb and Agg come from circuit complexity classes!
- We simplify presentation and write feature vectors in $\mathbb{R}$ instead of $\mathbb{R}^*$.

Gates make a circuit

- A gate: An entity that takes a finite sequence of input values returns an output:
  - Fan-in: How many inputs the gate takes.
  - Sum gate: $f_+(x, y) = x + y$ (fan-in 2), $f_+(x, y, y) = x + y + z$ (fan-in 3).
  - Product gate: $f(x, y) = x \times y$ (fan-in 2), $f(x_1, \ldots, x_n) = x_1 \times \cdots \times x_n$ (fan-in n).
  - In general, any n-ary function can be made a gate.

Input gates and output gates

- Put together a bunch of gates and you have a circuit.
- More gates:
  - Constant gate: 7 (fan-in 0).
  - Input gate: x (fan-in 0, labelled with a variable).
  - Output gate: Fan-in 1, receives the output of the circuit.

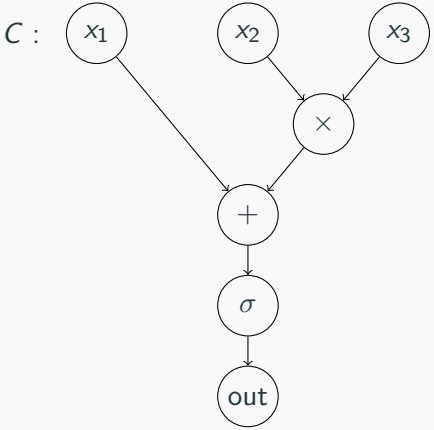
Input gates and output gates

- Put together a bunch of gates and you have a circuit.
- More gates:
  - Constant gate: 7 (fan-in 0).
  - Input gate: x (fan-in 0, labelled with a variable).
  - Output gate: Fan-in 1, receives the output of the circuit.
- A circuit computes a function:
  - Input is placed to the input gates.
  - Output is read from the output gate.

Arithmetic Circuits

Definition 11

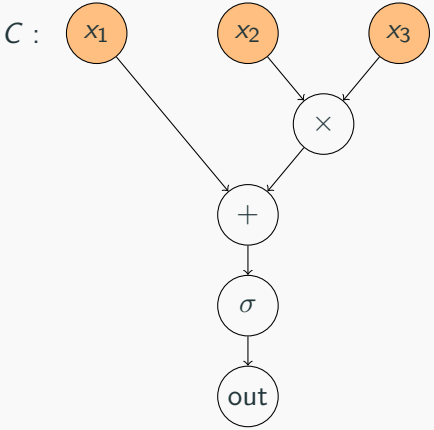
Directed acyclic graph with gates that perform arithmetic operations



Arithmetic Circuits

Definition 11

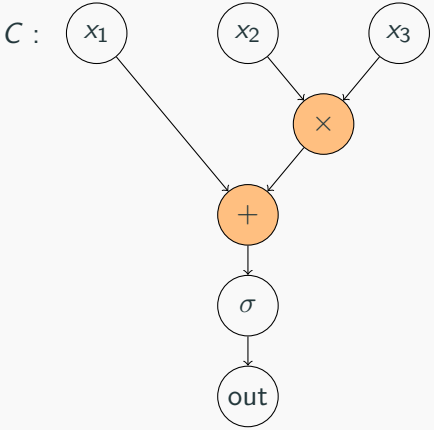
Directed acyclic graph with gates that perform arithmetic operations



Arithmetic Circuits

Definition 11

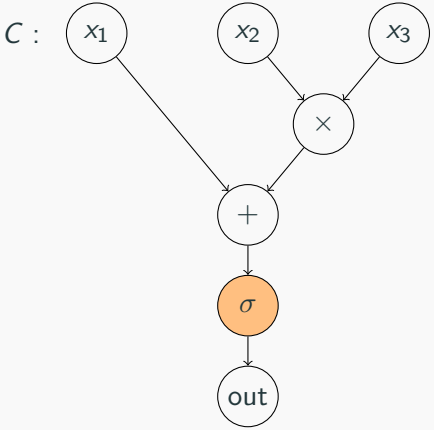
Directed acyclic graph with gates that perform arithmetic operations



Arithmetic Circuits

Definition 11

Directed acyclic graph with gates that perform arithmetic operations

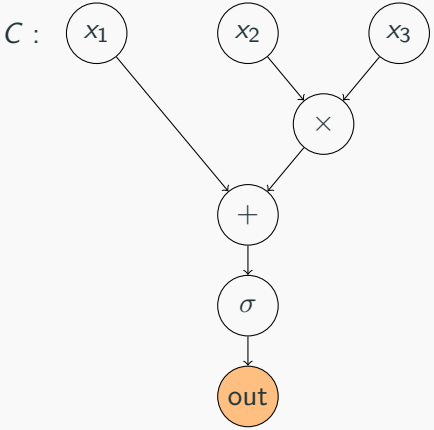


- additional function gates for activation functions

Arithmetic Circuits

Definition 11

Directed acyclic graph with gates that perform arithmetic operations

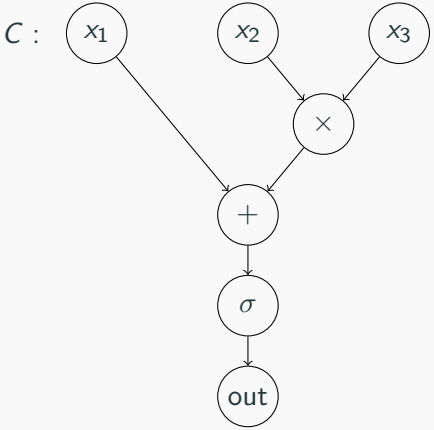


- $f_C(x_1, x_2, x_3) = \sigma(x_1 + x_2 x_3)$

Arithmetic Circuits

Definition 11

Directed acyclic graph with gates that perform arithmetic operations

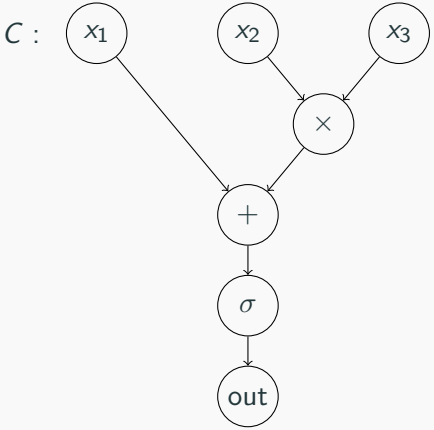


- size: number of gates
- depth: length of longest path from input to output

Arithmetic Circuits

Definition 11

Directed acyclic graph with gates that perform arithmetic operations

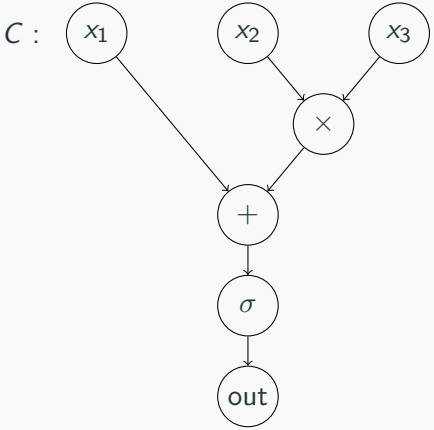


- size: number of gates
- depth: length of longest path from input to output
- A circuit has fixed number of input gates and computes $f : \mathbb{R}^p \to \mathbb{R}$.

Arithmetic Circuits

Definition 11

Directed acyclic graph with gates that perform arithmetic operations

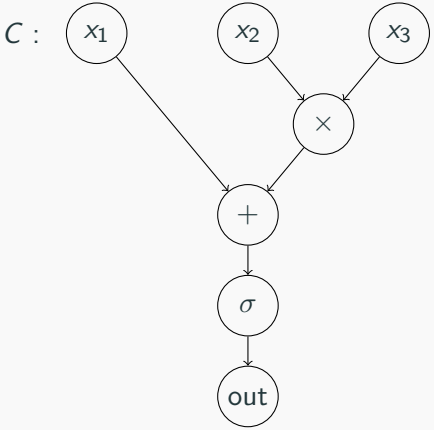


- size: number of gates
- depth: length of longest path from input to output
- A circuit has fixed number of input gates and computes $f : \mathbb{R}^p \to \mathbb{R}$.
- A circuit family $\{C_i\}_{i \in \mathbb{N}}$ has one circuit for each input length, and computes a function $f : \mathbb{R}^* \to \mathbb{R}$.

$\mathbb{R}^k$- Arithmetic Circuits

Definition 11

Directed acyclic graph with gates that perform arithmetic operations over $\mathbb{R}^k$.

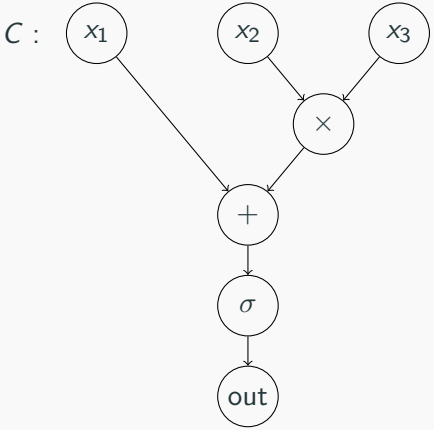


- size: number of gates
- depth: length of longest path from input to output
- A circuit has fixed number of input gates and computes $f : \mathbb{R}^p \to \mathbb{R}$.
- A circuit family $\{C_i\}_{i \in \mathbb{N}}$ has one circuit for each input length, and computes a function $f : \mathbb{R}^* \to \mathbb{R}$.
- Circuit function complexity classes obtained by restriction the size of C_i.

Arithmetic Circuits

Definition 11

Directed acyclic graph with gates that perform arithmetic operations over $\mathbb{R}^k$.



- size: number of gates
- depth: length of longest path from input to output
- A circuit family $\{C_i\}_{i\in\mathbb{N}}$ has one circuit for each input length, and computes a function $f : \mathbb{R}^* \to \mathbb{R}$.
- Circuit function complexity classes obtained by restriction the size of C_i.
  - $\mathrm{FAC}^0_{\mathbb{R}^k}$: constant depth/polynomial size
  - $\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$: additional function gates

Boolean Circuit complexity classes

- A Boolean circuit family $\{C_i\}_{i \in \mathbb{N}}$ computes a function $f : \mathbb{B}^* \to \mathbb{B}*$ such that

$$f(\vec{x}) \coloneqq C_{|\vec{x}|}(\vec{x}).$$

- Boolean circuits: strong connections to complexity theory
  - $+$ is an or-gate, $\times$ is an and-gate.
  - (uniform)-AC^0 is a well understood tractable complexity class.
    Problems that can be decided with constant depth polynomial size circuit families.
    Cannot express parity or majority.

Arithmetic Circuit complexity classes

- An arithmetic circuit family $\{C_i\}_{i \in \mathbb{N}}$ computes a function $f : \mathbb{R}^* \to \mathbb{R}^*$ such that

$$f(\vec{x}) := C_{|\vec{x}|}(\vec{x}).$$

- Arithmetic circuits: function complexity classes
  - Characterise: Which functions $f : \mathbb{R}^* \to \mathbb{R}^*$ can be computed with circuit families.
  - $\mathrm{FAC}^0_{\mathbb{R}}$: functions by circuit families of constant depth and polynomial size.
  - Less studied than Boolean circuits, but many results known:
    - a polynomial of degree 2^{2^n} require a circuit of size 2^n.
    - determinant of an $n \times n$ matrix: a circuit of polynomial size and $\mathcal{O}(\log^2(n))$ depth.

Arithmetic Circuit complexity classes

- An arithmetic circuit family $\{C_i\}_{i \in \mathbb{N}}$ computes a function $f : \mathbb{R}^* \to \mathbb{R}^*$ such that

$$f(\vec{x}) := C_{|\vec{x}|}(\vec{x}).$$

- Arithmetic circuits: function complexity classes
  - Characterise: Which functions $f : \mathbb{R}^* \to \mathbb{R}^*$ can be computed with circuit families.
  - $\mathrm{FAC}^0_{\mathbb{R}}$: functions by circuit families of constant depth and polynomial size.
  - Less studied than Boolean circuits, but many results known:
    - a polynomial of degree 2^{2^n} require a circuit of size 2^n.
    - determinant of an $n \times n$ matrix: a circuit of polynomial size and $\mathcal{O}(\log^2(n))$ depth.
- Question: Can we find precise correspondence between expressivity of GNNs and circuit complexity classes?

Examples of circuit families

- The sum aggregation $\sum_{x \in E(n)} g(x)$ is a very simple circuit family of depth 1.
- Given a specification of aggregation and combination function in terms of $+$, x, and activation functions σ_i, it is easy to define the corresponding circuit class with gates for $+$, x, and σ_i.
- Most common aggregation and combination function used in practice can be computed with constant depth polynomial size circuit families.

Circuits, circuit families, and GNNs

- A feedforward neural network is an arithmetic circuit of very regular form!
- Gates for scalar multiplication, addition, and for the activation function.

Circuits, circuit families, and GNNs

- A feedforward neural network is an arithmetic circuit of very regular form!
- Gates for scalar multiplication, addition, and for the activation function.
- A GNN-layer is a very regular circuit family!
- A node of degree i uses the circuit C_{i+1}.

Circuits, circuit families, and GNNs

- A feedforward neural network is an arithmetic circuit of very regular form!
- Gates for scalar multiplication, addition, and for the activation function.
- A GNN-layer is a very regular circuit family!
- A node of degree i uses the circuit C_{i+1}.
- A graph transformation that a GNN computes is a concatanation of it's layers $\Rightarrow$ a GNN is just a special kind of circuit family.

Circuits, circuit families, and GNNs

- A feedforward neural network is an arithmetic circuit of very regular form!
- Gates for scalar multiplication, addition, and for the activation function.
- A GNN-layer is a very regular circuit family!
- A node of degree i uses the circuit C_{i+1}.
- A graph transformation that a GNN computes is a concatanation of it's layers $\Rightarrow$ a GNN is just a special kind of circuit family.
- Why is this interesting?
  - Relating GNNs with artihmetic circuit complexity classes, gives a mature way to understand capabilites of GNNs.

Arithmetic circuits and GNNs

1st Goal: Show that GNNs can be simulated by circuit families

Let N be a GNN. Then there exists a circuit family $\mathcal{C}$, such that for all labeled graphs the circuit family computes the same feature vectors as N.

2nd Goal: Show that circuit families can be simulated by GNNs

Let $\mathcal{C}$ be a circuit family. Show there exists a GNN N that has the output of the circuit family among its feature vectors.

Goal 1: Simulate GNNs with circuits

1st Goal: Show that GNNs can be simulated by circuit families

Let N be a GNN. Then there exists a circuit family $\mathcal{C}$, such that for all labeled graphs the circuit family computes the same feature vectors as N.

- Input graphs G for GNNs are encoded as as strings of real numbers by listing its adjacency matrix and feature vectors $\langle G \rangle$.
- Circuit family simulates the GNN such that $N(G) = G'$ iff $\mathcal{C}(\langle G \rangle) = \langle G' \rangle$.

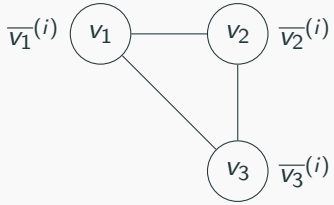
Goal 2: Simulate circuits with GNNs

2nd Goal: Show that circuit families can be simulated by GNNs

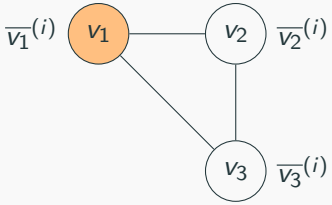
Let $\mathcal{C}$ be a circuit family. Show there exists a GNN N that has the output of the circuit family among its feature vectors.

- GNN receives the graph structure and input of the circuit as the input graph

Example: AC-GNN

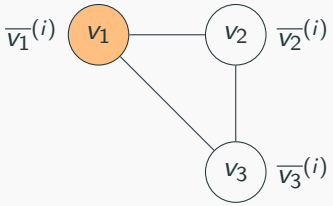


Example: AC-GNN



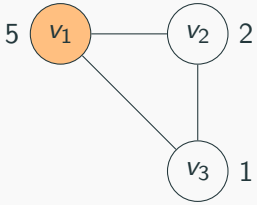
$$\overline{v_1}^{(i+1)} = \mathsf{COM}^{(i)}(\overline{v_1}^{(i)}, \mathsf{AGG}^{(i)}(\overline{v_2}^{(i)}, \overline{v_3}^{(i)}))$$

Example: AC-GNN



$$\overline{v_1}^{(i+1)} = \mathrm{sigmoid}(\mathrm{avg}(\overline{v_1}^{(i)}, (\overline{v_2}^{(i)} + \overline{v_3}^{(i)})))$$

Example: AC-GNN

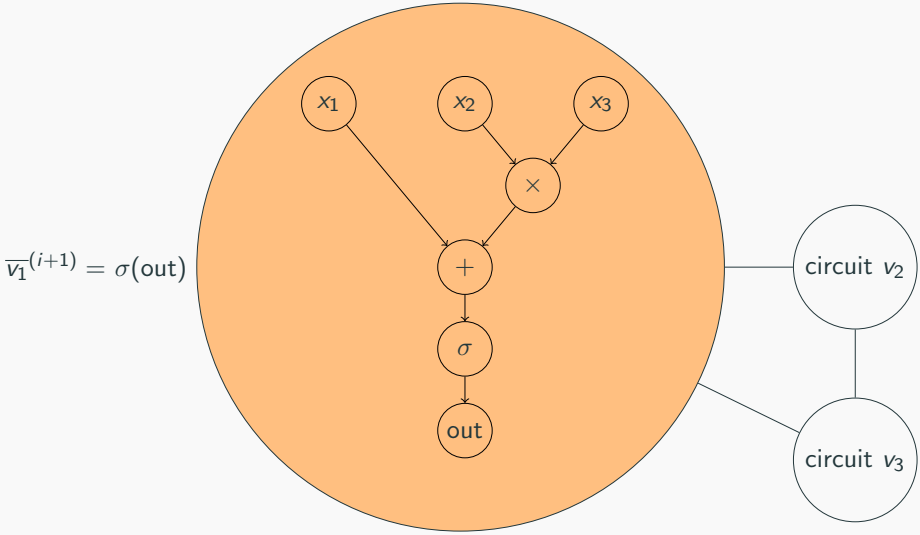


$$\overline{v_1}^{(i+1)} = \mathrm{sigmoid}(\mathrm{avg}(5, (2+1)))$$

Circuit GNNs: An abstraction of AC-GNNs

- Goal: Abstract concrete aggregation-combination functions to ones that come from a circuit function class.
- Why? Facilitate clean and general statements for capabilities for range of GNNs.

Example: Circuit-GNN



Building blocks of Circuit-GNNs

- Goal: Define GNNs in a level of abstraction that makes relating to circuits easier.
- Aggregate-combine layer will be defined using circuits.

Building blocks of Circuit-GNNs

- Goal: Define GNNs in a level of abstraction that makes relating to circuits easier.
- Aggregate-combine layer will be defined using circuits.
- Agg-Comb basis of a circuit GNN:
  - Fix $\mathcal{S}$, a set of functions $\mathbb{R}^* \to \mathbb{R}$ (aggregate-combine layer).
  - Fix $\mathcal{A}$, a set of activation functions $\mathbb{R} \to \mathbb{R}$ (activation function of a layer).

Building blocks of Circuit-GNNs

- Goal: Define GNNs in a level of abstraction that makes relating to circuits easier.
- Aggregate-combine layer will be defined using circuits.
- Agg-Comb basis of a circuit GNN:
  - Fix $\mathcal{S}$, a set of functions $\mathbb{R}^* \to \mathbb{R}$ (aggregate-combine layer).
  - Fix $\mathcal{A}$, a set of activation functions $\mathbb{R} \to \mathbb{R}$ (activation function of a layer).
- $(\mathcal{S}, \mathcal{A})$-GNN of depth d is a function $[d] \to \mathcal{S} \times \mathcal{A}$.

Building blocks of Circuit-GNNs

- Goal: Define GNNs in a level of abstraction that makes relating to circuits easier.
- Aggregate-combine layer will be defined using circuits.
- Agg-Comb basis of a circuit GNN:
  - Fix $\mathcal{S}$, a set of functions $\mathbb{R}^* \to \mathbb{R}$ (aggregate-combine layer).
  - Fix $\mathcal{A}$, a set of activation functions $\mathbb{R} \to \mathbb{R}$ (activation function of a layer).
- $(\mathcal{S}, \mathcal{A})$-GNN of depth d is a function $[d] \to \mathcal{S} \times \mathcal{A}$.
  - Standard $\mathrm{Comb}^i\Big(g(n), \mathrm{Agg}^i\big(\{\{g(m) \mid m \in E(n)\}\}\big)\Big)$ is replaced with applying $(f_i, \sigma_i) \in \mathcal{S} \times \mathcal{A}$ given by the function d.

Building blocks of Circuit-GNNs

- Goal: Define GNNs in a level of abstraction that makes relating to circuits easier.
- Aggregate-combine layer will be defined using circuits.
- Agg-Comb basis of a circuit GNN:
  - Fix $\mathcal{S}$, a set of functions $\mathbb{R}^* \to \mathbb{R}$ (aggregate-combine layer).
  - Fix $\mathcal{A}$, a set of activation functions $\mathbb{R} \to \mathbb{R}$ (activation function of a layer).
- $(\mathcal{S}, \mathcal{A})$-GNN of depth d is a function $[d] \to \mathcal{S} \times \mathcal{A}$.
  - Standard $\mathrm{Comb}^i\Big(g(n), \mathrm{Agg}^i\big(\{\{g(m) \mid m \in E(n)\}\}\big)\Big)$ is replaced with applying $(f_i, \sigma_i) \in \mathcal{S} \times \mathcal{A}$ given by the function d.
- if all functions in $\mathcal{S}$ belong to circuit function class $\mathfrak{F}$: $(\mathfrak{F}, \mathcal{A})$-GNN, e.g. $(\mathrm{FAC}^0_{\mathbb{R}}, \{\mathrm{id}\})$-GNN

Building blocks of Circuit-GNNs

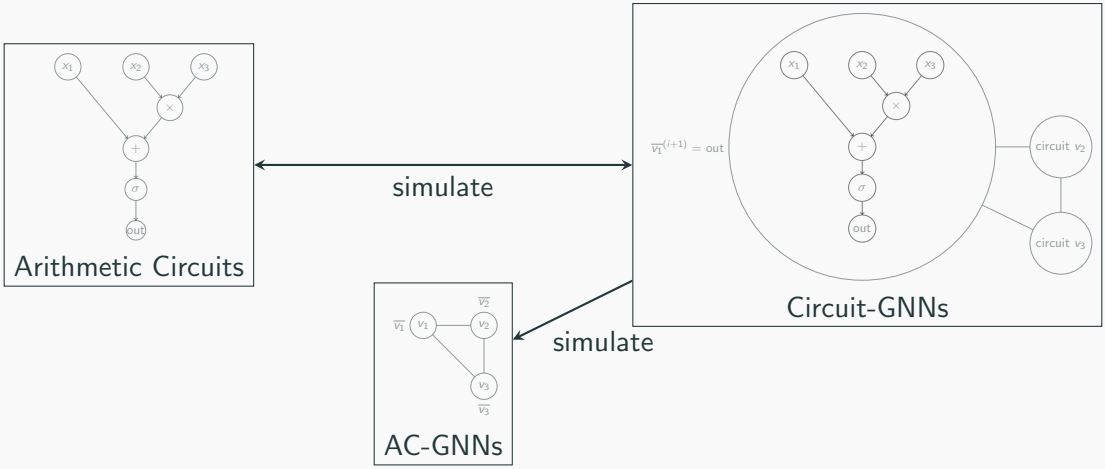
- Goal: Define GNNs in a level of abstraction that makes relating to circuits easier.
- Aggregate-combine layer will be defined using circuits.
- Agg-Comb basis of a circuit GNN:
  - Fix $\mathcal{S}$, a set of functions $\mathbb{R}^* \to \mathbb{R}$ (aggregate-combine layer).
  - Fix $\mathcal{A}$, a set of activation functions $\mathbb{R} \to \mathbb{R}$ (activation function of a layer).
- $(\mathcal{S}, \mathcal{A})$-GNN of depth d is a function $[d] \to \mathcal{S} \times \mathcal{A}$.
  - Standard $\mathrm{Comb}^i\Big(g(n), \mathrm{Agg}^i\big(\{\{g(m) \mid m \in E(n)\}\}\big)\Big)$ is replaced with applying $(f_i, \sigma_i) \in \mathcal{S} \times \mathcal{A}$ given by the function d.
- if all functions in $\mathcal{S}$ belong to circuit function class $\mathfrak{F}$: $(\mathfrak{F}, \mathcal{A})$-GNN, e.g. $(\mathrm{FAC}^0_{\mathbb{R}}, \{\mathrm{id}\})$-GNN

Intuition: a GNN with circuits in its nodes

Circuit GNNs

- Circuit GNNs are AC-GNNs where complexity of layer computation can be specified using circuit complexity theory!

Overview of Connections



AC-GNNs as C-GNNs

- For any AC-GNN with activation functions $\mathcal{A}$ and where the aggregation functions are computable by $\mathrm{FAC}^0_{\mathbb{R}}$-circuit families, there exists a $\big(\mathrm{FAC}^0_{\mathbb{R}}, \mathcal{A}\big)$-GNN with a constant number of layers which computes the same function, omitting only the functionality of the classification function.
- This holds for the common aggregation functions like the sum, product or mean.

Simulating Circuit-GNNs with Arithmetic Circuits

A function is tail symmetric, if $f(x_1, y_1, \ldots, y_n) = f(x_1, p(y_1), \ldots, p(y_n))$, for all variable permutations p.

Theorem 12

Let N be a $\left(t\mathrm{FAC}^0_{\mathbb{R}}, \mathcal{A}\right)$-GNN. Then there exists an $\mathrm{FAC}^0_{\mathbb{R}}[\mathcal{A}]$-circuit family $\mathcal{C}$, such that for all labeled graphs G the circuit family computes the same feature vectors as N.

Simulating Circuit-GNNs with Arithmetic Circuits

A function is tail symmetric, if $f(x_1, y_1, \ldots, y_n) = f(x_1, p(y_1), \ldots, p(y_n))$, for all variable permutations p.

Theorem 12

Let N be a $\left(t\mathrm{FAC}^0_{\mathbb{R}}, \mathcal{A}\right)$-GNN. Then there exists an $\mathrm{FAC}^0_{\mathbb{R}}[\mathcal{A}]$-circuit family $\mathcal{C}$, such that for all labeled graphs G the circuit family computes the same feature vectors as N.

- Each GNN-layer is computed by a $\mathrm{FAC}^0_{\mathbb{R}}[\mathcal{A}]$-circuit family
- For any input graph G, a node $v \in V$, and layer i, the feature vector $v^{(i+1)}$ is computed via a suitable circuit from the fixed circuit families.
- Essentially $N(G)$ computation unrolled is a big $\mathrm{FAC}^0_{\mathbb{R}}[\mathcal{A}]$-circuit.
- Difficulty: Construct a circuit family $\mathcal{C}$ such that $C_{|\langle G \rangle|} = \langle N(G) \rangle$ for all G. (output the graph G as adjacency matrix and corresponding feature vectors)
  - The idea is to construct the circuits so that it uses the input adjacency matrix to select suitable circuits from the layer families to compute with.
  - C_k contains all layer circuits needed for degree k graphs.

Simulating Arithmetic Circuits with Circuit-GNNs

Theorem

For every $\mathrm{FAC}^0_{\mathbb{R}}[\mathcal{A}]$-circuit family $\mathcal{C}$ there exists a $\left(t\mathrm{FAC}^0_{\mathbb{R}}[\mathcal{A}], \{\mathrm{id}\}\right)$-GNN N such that for all $n \in \mathbb{N}$ and $\vec{x} \in \mathbb{R}^n$ $N(G_{C_n,\vec{x}})$ has $C_n(\vec{x})$ as feature vectors.

- the graph structure of the circuit is given as the input graph of the Circuit-GNN
- Pre-process the circuit to path-length normal form.
  - All paths from an input to an output gate has the same length.
  - Every non-input and non-output gate has exactly one successor.
- simulate circuit computation layerwise and store intermediate results in features
- Difficulty: A circuit computes a gate when inputs to it are computed, but a GNN re-computes all feature values on every layer.
  - Solution: Add an additional feature value that is used to guide the GNN computation by resetting feature values so that circuit computation can be simulated.

Simulating Arithmetic Circuits with Circuit-GNNs

A circuit is in function layer form, if it is in path-length normal form, and on each depth the type of gate is uniform.

Theorem

For every $f\mathrm{FAC}^0_{\mathbb{R}}[\mathcal{A}]$-circuit family $\mathcal{C}$ there exists a $\left(t\mathrm{FAC}^0_{\mathbb{R}}, \mathcal{A} \cup \{\mathrm{id}\}\right)$-GNN N such that for all $n \in \mathbb{N}$ and $\vec{x} \in \mathbb{R}^n$ $N(G_{C_n,\vec{x}})$ has $C_n(\vec{x})$ as feature vectors.

- the graph structure of the circuit is given as the input graph of the Circuit-GNN
- Pre-process the circuit to path-length normal form.
  - All paths from an input to an output gate has the same length.
  - Every non-input and non-output gate has exactly one successor.
- simulate circuit computation layerwise and store intermediate results in features
- Difficulty: A circuit computes a gate when inputs to it are computed, but a GNN re-computes all feature values on every layer.
  - Solution: Add an additional feature value that is used to guide the GNN computation by resetting feature values so that circuit computation can be simulated.

Can we make this recursive?

Definition 13

An L layer *aggregate combine graph neural network* (AC-GNN) is a tuple $\mathcal{D} = (\{\mathsf{AGG}^{(i)}\}_{i=1}^{L}, \{\mathsf{COM}^{(i)}\}_{i=1}^{L}, \{\sigma^{(i)}\}_{i=1}^{L}, \mathsf{CLS})$, where

- $\{\mathsf{AGG}^{(i)}\}_{i=1}^{L}$: aggregate functions
- $\{\mathsf{COM}^{(i)}\}_{i=1}^{L}$: combine functions
- $\{\sigma^{(i)}\}_{i=1}^{L}$: activation functions
- CLS: $\mathbb{R} \to \{0, 1\}$: classification function

Updating vectors in every layer $1 \leq i \leq L$ as follows:

- initial feature vector of v: $\overline{v}^{(0)} = g_V(v)$
- for $i > 1$:
  - $\overline{v}^{(i)} = \sigma^{(i)}\left(\mathsf{COM}^{(i)}\left(\overline{v}^{(i-1)}, \overline{y}\right)\right)$, where $\overline{y} = \mathsf{AGG}^{(i)}\left(\{\{\overline{u}^{(i-1)} \mid u \in N_G(v)\}\}\right)$

Definition 14

A period length d *recurrent aggregate combine graph neural network* (AC-GNN) is a tuple $\mathcal{D} = (\{\mathrm{AGG}^{(i)}\}_{i=1}^{d}, \{\mathrm{COM}^{(i)}\}_{i=1}^{d}, \{\sigma^{(i)}\}_{i=1}^{d}, \mathit{HLT}, \mathrm{CLS})$, where

- $\{\mathrm{AGG}^{(i)}\}_{i=1}^{d}$: aggregate functions
- $\{\mathrm{COM}^{(i)}\}_{i=1}^{d}$: combine functions
- $\{\sigma^{(i)}\}_{i=1}^{d}$: activation functions
- $\mathrm{HLT}\colon \mathbb{N}_{>0} \times \mathbb{R}^{*} \to \{0,1\}$
- $\mathrm{CLS}\colon \mathbb{R} \to \{0,1\}$: classification function

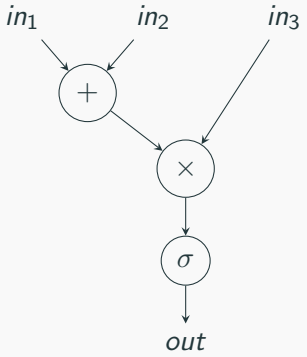
Updating vectors in every layer $1 \leq i \leq d$ as follows:

- initial feature vector of v: $\overline{v}^{(0)} = g_V(v)$
- for $i > 1$:
  - $\overline{v}^{(i)} = \sigma^{(i)}\left(\mathrm{COM}^{(i)}\left(\overline{v}^{(i-1)}, \overline{y}\right)\right)$, where $\overline{y} = \mathrm{AGG}^{(i)}\left(\{\{\overline{u}^{(i-1)} \mid u \in N_G(v)\}\}\right)$
  - $\mathrm{HLT}\left(i, \left(x_v^{(i)} \mid v \in V\right)\right)$

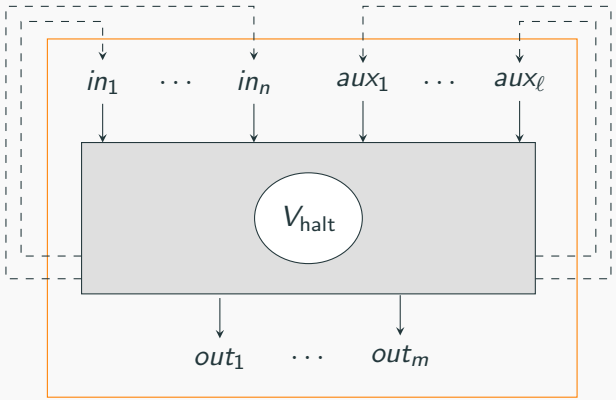
Arithmetic Circuits

Definition 15

Directed acyclic graph with gates that perform arithmetic operations over $\mathbb{R}$.



Recurrent Arithmetic Circuit



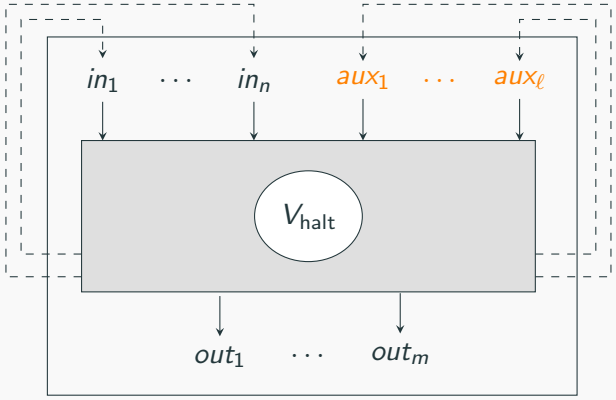
underlying extended arithmetic circuit C

Definition 16

rec-$C = (C, \bar{a}, E_{\text{rec}}, \text{HLT}, V_{\text{halt}})$

- $\text{HLT}\colon \mathbb{N}_{>0} \times \mathbb{R}^* \to \{0, 1\}$

Recurrent Arithmetic Circuit



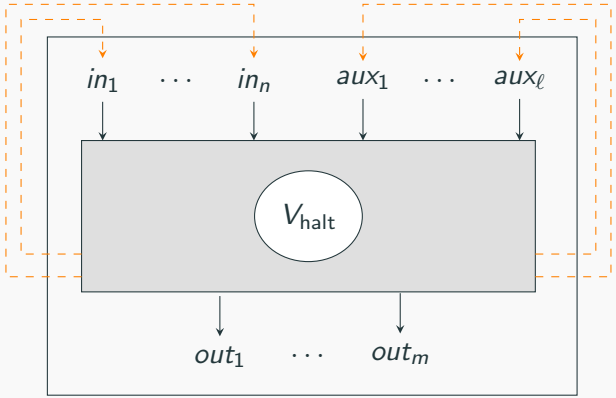
underlying extended arithmetic circuit C

Definition 16

rec-$C = (C, \overline{a}, E_{\text{rec}}, \text{HLT}, V_{\text{halt}})$

- $\text{HLT}\colon \mathbb{N}_{>0} \times \mathbb{R}^* \to \{0, 1\}$

Recurrent Arithmetic Circuit



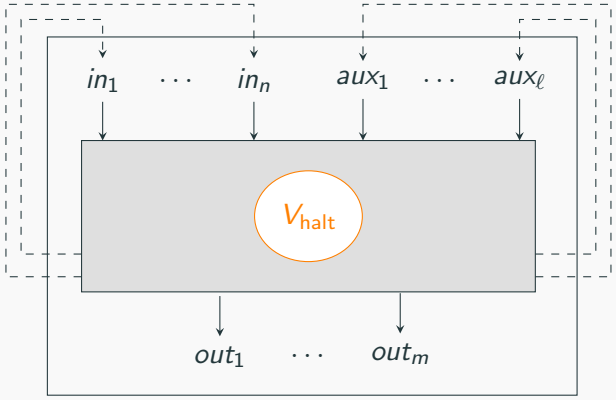
underlying extended arithmetic circuit C

Definition 16

rec-$C = (C, \bar{a}, E_{\text{rec}}, \text{HLT}, V_{\text{halt}})$

- $\text{HLT}\colon \mathbb{N}_{>0} \times \mathbb{R}^* \to \{0, 1\}$

Recurrent Arithmetic Circuit



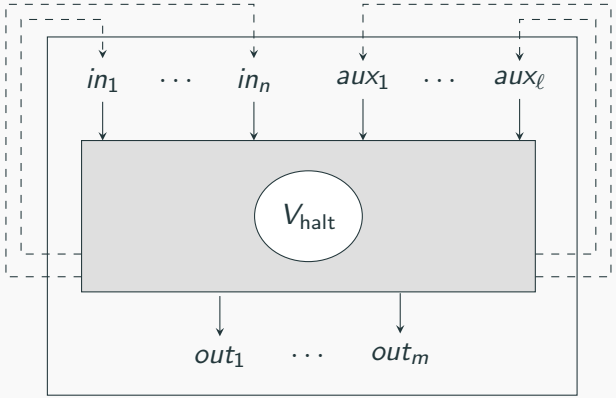
underlying extended arithmetic circuit C

Definition 16

rec-$C = (C, \bar{a}, E_{\text{rec}}, \text{HLT}, V_{\text{halt}})$

- $\text{HLT}\colon \mathbb{N}_{>0} \times \mathbb{R}^* \to \{0, 1\}$

Recurrent Arithmetic Circuit



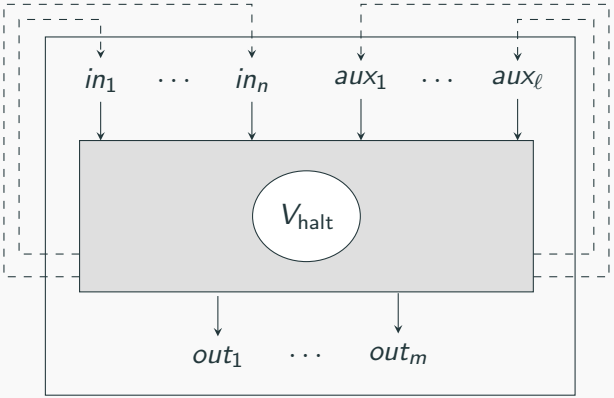
underlying extended arithmetic circuit C

Definition 16

rec-$C = (C, \bar{a}, E_{\text{rec}}, \text{HLT}, V_{\text{halt}})$

- $\text{HLT} \colon \mathbb{N}_{>0} \times \mathbb{R}^* \to \{0, 1\}$

Recurrent Arithmetic Circuit



underlying extended arithmetic circuit C

Definition 16

rec-$C = (C, \overline{a}, E_{\text{rec}}, \text{HLT}, V_{\text{halt}})$

- $\text{HLT}\colon \mathbb{N}_{>0} \times \mathbb{R}^* \to \{0,1\}$

Recurrent Circuit Function Class

$f_{\text{rec-}\mathcal{C}}\colon \mathbb{R}^* \to \mathbb{R}^* \in \text{rec}[\mathfrak{F}_{\text{halt}}]\text{-}\mathfrak{F}_C$ if:

- function of underlying arithmetic circuit family $\in \mathfrak{F}_C$
- halting functions $\in \mathfrak{F}_{\text{halt}}$

Example: Fibonacci sequence: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$

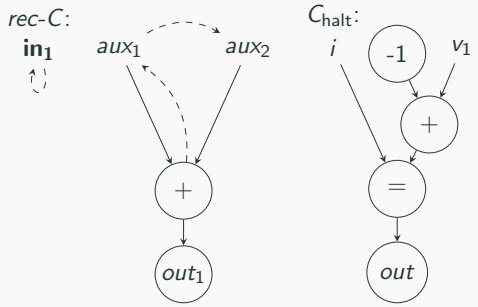
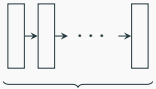


Figure 1: Recurrent circuit *rec-C* with halting circuit C_{halt} computing the x_1-th Fibonacci number for $1 < x_1 \in \mathbb{N}$. The dashed arrows mark recurrent edges and the bold gate in_1 is the single halting gate, i. e. the gate whose value forms the input for C_{halt} along with the iteration number i. aux_1 and aux_2 originally store F_0 and F_1.

Recursive GNNs and Recursive circuits

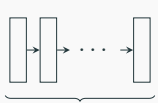
- As we saw: Arithmetic circuits characterise non-recursive GNNs.
- Recursive circuits do the same for recursive GNNs.
- For this we introduce recursive circuit GNNs.
- rec$[t\mathfrak{F}_{\text{halt}}]$-$(t\mathfrak{F}, \mathcal{A})$-GNN:
  - Halting Complexity in $t\mathfrak{F}_{\text{halt}}$
  - Layer Computation Complexity in $t\mathfrak{F}$

Models of Recurrent C-GNNs

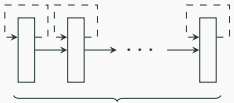


(a) C-GNN without recurrence and a fixed depth $d \in \mathbb{N}$

Models of Recurrent C-GNNs

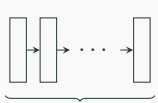


(a) C-GNN without recurrence and a fixed depth $d \in \mathbb{N}$

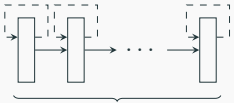


(b) C-GNN with only inner recurrence and a fixed depth $d \in \mathbb{N}$

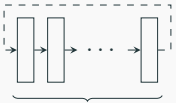
Models of Recurrent C-GNNs



(a) C-GNN without recurrence and a fixed depth $d \in \mathbb{N}$

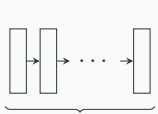


(b) C-GNN with only inner recurrence and a fixed depth $d \in \mathbb{N}$

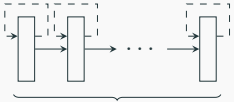


(c) C-GNN with only outer recurrence

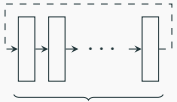
Models of Recurrent C-GNNs



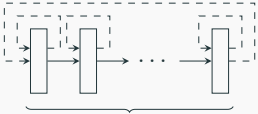
(a) C-GNN without recurrence and a fixed depth $d \in \mathbb{N}$



(b) C-GNN with only inner recurrence and a fixed depth $d \in \mathbb{N}$

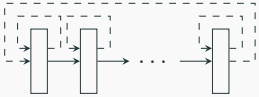


(c) C-GNN with only outer recurrence

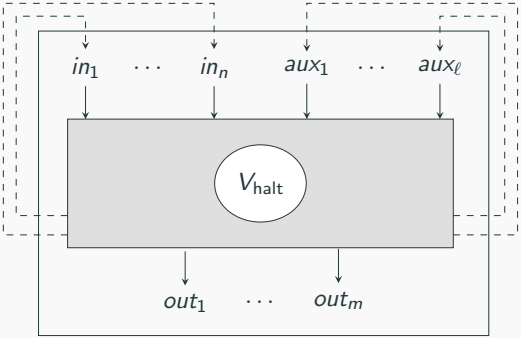


(d) C-GNNs with inner and outer recurrence

Simulating Recurrent C-GNNs With Recurrent Arithmetic Circuits



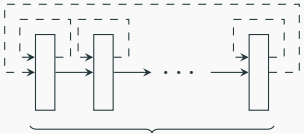
Recurrent C-GNN



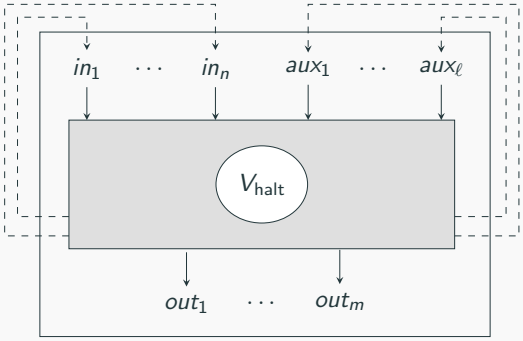
Recurrent Arithmetic Circuit

Simulating Recurrent C-GNNs With Recurrent Arithmetic Circuits

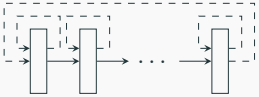
- Encode labelled graphs and use them as input to the circuit as before
- Simulate the layers of the recurrent C-GNNs as before
- New: Use the recurrence of the circuit to simulate recurrence of the C-GNN



Simulating Recurrent Arithmetic Circuits with Recurrent C-GNNs

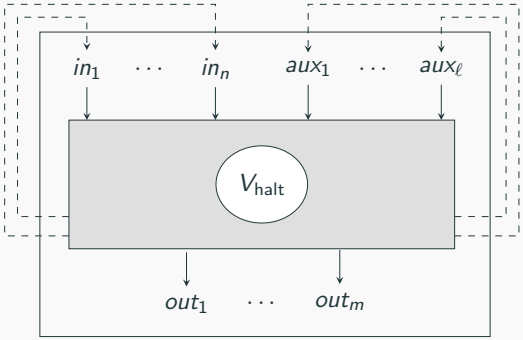


Recurrent Arithmetic Circuit

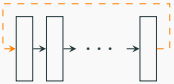


Recurrent C-GNN

Simulating Recurrent Arithmetic Circuits with Outer Recurrent C-GNNs



Recurrent Arithmetic Circuit



Outer Recurrent C-GNN

Simulating Recurrent Arithmetic Circuits with Outer Recurrent C-GNNs

Idea: Encode a recurrent arithmetic circuit as a labelled graph and define a recurrent C-GNN that works on this labelled graph as an input

- Fix a notion of logspace labelled graph encoding of a recurrent circuit
- Simulate the computations of the recurrent circuit
- Halting function of the circuit needs to be partially simulated partly as well
- Challenge: GNN feature vectors need to resetted to simulate recursive computation of the circuit.

Theorem 17

Let $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ be a rec$[\mathfrak{F}]$-$\mathfrak{F}[\mathcal{A}]$-circuit family. Then there exists a rec$[\mathrm{t}\mathfrak{F}]$-$(\mathrm{t}\mathfrak{F}[\mathcal{A}], \{id\})$-GNN, such that such that for all $n \in \mathbb{N}$ and $\vec{x} \in \mathbb{R}^n$ $N(G_{C_n,\vec{x}})$ has $C_n(\vec{x})$ as feature vectors.

Simulating Recurrent Arithmetic Circuits with Outer Recurrent C-GNNs

Idea: Encode a recurrent arithmetic circuit as a labelled graph and define a recurrent C-GNN that works on this labelled graph as an input

- Fix a notion of logspace labelled graph encoding of a recurrent circuit
- Simulate the computations of the recurrent circuit
- Halting function of the circuit needs to be partially simulated partly as well
- Challenge: GNN feature vectors need to resetted to simulate recursive computation of the circuit.

As before, a stronger result holds for circuits in function layer form.

Theorem 17

Let $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ be a rec$[\mathfrak{F}]$-$f\mathfrak{F}[\mathcal{A}]$-circuit family. Then there exists a rec$[\mathfrak{tF}]$-$(\mathfrak{tF}, \mathcal{A}\cup\{id\})$-GNN, such that such that for all $n \in \mathbb{N}$ and $\vec{x} \in \mathbb{R}^n$ $N(G_{C_n,\vec{x}})$ has $C_n(\vec{x})$ as feature vectors.

Conclusion

- Expressivity of GNNs can be rigorously studied via arithmetic circuits.
- Expressivity of recursive GNNs can be rigorously studied via recursive arithmetic circuits.
- Deep connections between circuit complexity classes and problems decidable with GNNs.

Bibliography i

[Mor+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: *AAAI*. 2019.