Lecture 3: Basics of recurrent GNNs, μ-calculus, and their uniform expressivity

Roadmap

- We now move from fixed-depth GNNs to recurrent GNNs.
- We revise the difference between:

  distinguishing power and uniform expressivity.

- We show that recurrent GNNs do not add distinguishing power.
- But they do add uniform expressive power! yes!
- Example: reachability.

Part 5: GNNs and recursion

- What do we cover:
  - Different approaches to add recursion to GNNs.
  - Expressivity of these recurrent GNN alternatives.
  - Two extension of GML to define recursive classifiers.
- Main sources
  - Veeti Ahvonen, Damian Heiman, Antti Kuusisto, and Carsten Lutz. Logical Characterizations of Recurrent Graph Neural Networks with Reals and Floats. Neurips 2024.
  - Jeroen Bollen, Jan Van den Bussche, Stijn Vansummeren, Jonni Virtema. Halting Recurrent GNNs and the Graded μ-Calculus. KR 2025.
  - Eran Rosenbluth, Martin Grohe. Repetition Makes Perfect: Recurrent Graph Neural Networks Match Message-Passing Limit. AAAI 2026.

Recurrent GNNs

Recurrent = Keep repeating GNN layers.

- In fact, the original GNNs in the paper by Scarselli, Gori, Tsoi, Hagenbuchner, Monfardini (*IEEE TNN* 2009) were recurrent!
- They define a vertex feature update via a global transition function $F_{\boldsymbol{\theta}}$:

$$g_{t+1}(n) = F_{\boldsymbol{\theta}}\big(g_t(n)\big).$$

Recurrent GNNs

How are these trained?

- Banach fixed-point theorem: if $F_{\boldsymbol{\theta}}$ is a contraction, i.e. $\exists\, \mu < 1$ with $\|F_{\boldsymbol{\theta}}(x) - F_{\boldsymbol{\theta}}(y)\| \leq \mu \|x - y\|$ (for all x, y) then a unique fixed point x^* exists, and the iteration above converges to it exponentially fast from any x_0.

- **Training** (gradient descent):
  1. iterate $g_{t+1} = F_{\boldsymbol{\theta}}(g_t)$ until (approximately) converged to g^*;
  2. compute the loss and its gradient w.r.t. $\boldsymbol{\theta}$;
  3. update $\boldsymbol{\theta}$.

Recurrent GNNs: Challenging to model

Theoretically, it is hard to work with fixpoints of such contractions.

Also, how is convergence decided? How many steps.

If only up to finite iteration, why not use simple non-recurrent GNNs?

- There is no unique recurrent GNN model.
- Commonality: repeated message passing
- Differences: When the recurrent GNN is "done" with a node or graph.

Recurrent GNNs

Shared recurrence

$$g_t^0(n) = \mathsf{Init}(g(n)), \qquad g_t^{t+1}(n) = \mathsf{Comb}(g_t(n), \mathsf{Agg}\{\{g_t(m) \mid m \in E(n)\}\}) .$$

We denote by $\mathcal{L} = (\mathsf{Ini}, L)$ and $L = (\mathsf{Comb}, \mathsf{Agg})$ the recurrent layer.

Architectural choices

- **Domain:** reals, floats, rationals, bitstrings.
- **Aggregation:** sum, ordered float-sum, arbitrary multiset map.
- **Extra's:** graph size, global aggregation, random initialization.

Semantic choices

- **Stopping:** accepting vector, global halt, finished flag, fixed point.
- **Output:** node classifier vs. feature transformer.

Fixed-depth versus recurrent GNNs

- A fixed-depth GNN applies a fixed sequence of layers

$$L^{(1)}, \ldots, L^{(\ell)}.$$

- The number of message-passing rounds is fixed in advance.
- A recurrent GNN applies the same layer repeatedly:

$$G_0, \; G_1 = L(G_0), \; G_2 = L(G_1), \; \ldots$$

- The layer L may at times be composition of more single layers.
- The number of rounds may depend on the input graph.
- Thus recurrent GNNs are designed to express iterative or recursive computations.

Challenge of recursion

- From one layer to another recurrent GNNs behave as their non-recurrent counterparts.
- Given an labelled input graph, each layer computes a feature transformation

$$G[X] \to \mathcal{G}[Y]$$

via the standard equation

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

- Challenge: When do we read the output of the computation?
- We cover two alternative approaches.

Two alternative approaches

- Accepting state model
- Veeti Ahvonen, Damian Heiman, Antti Kuusisto, and Carsten Lutz. Logical Characterizations of Recurrent Graph Neural Networks with Reals and Floats. Neurips 2024.
- Halting model
- Jeroen Bollen, Jan Van den Bussche, Stijn Vansummeren, Jonni Virtema. Halting Recurrent GNNs and the Graded μ-Calculus. KR 2025.

Part 5 (a): GNNs and recursion: Accepting state model

- Veeti Ahvonen, Damian Heiman, Antti Kuusisto, and Carsten Lutz. Logical Characterizations of Recurrent Graph Neural Networks with Reals and Floats.

Accepting state model

Let X be a finite input label set.

- A recurrent GNN is a tuple
$$\mathcal{N} = (\mathsf{In}, L, \mathsf{Hlt}).$$
- The map
$$\mathsf{In} : X \to \mathbb{R}^d$$
initializes node features.
- The layer L is an aggregate-combine layer
$$L : \mathcal{G}[\mathbb{R}^d] \to \mathcal{G}[\mathbb{R}^d].$$
- The function
$$\mathsf{Hlt} : \mathbb{R}^d \to \mathbb{B}$$
decides whether a node is ready to halt.

Basic graph concepts: pointwise lifting

- Let
$$\alpha \colon X \to Y$$
be a function between label sets. We write
$$\alpha^{\uparrow} \colon \mathcal{G}[X] \to \mathcal{G}[Y]$$
for the pointwise lifting of α to labeled graphs.
- For an X-labeled graph
$$G = (N, E, g),$$
it is defined by
$$\alpha^{\uparrow}(G) := (N, E, \alpha \circ g).$$

Runs in the accepting state model

Let $G = (N, E, g)$ be an X-labeled graph.

- The initial graph is
$$G_0 \coloneqq \mathsf{In}^{\uparrow}(G).$$
- The recurrent layer is applied repeatedly:
$$G_{i+1} \coloneqq L(G_i).$$
- A run accepts (G, n) iff
$$\mathsf{Hlt}(g_k(n)) = 1 \text{ for some } k \in \mathbb{N},$$
with $G_k = (N, E, g_k)$.

Acceptance by halting at the node level.

Acceptance: the GNN decides itself

Acceptance feature vector f such that $\mathsf{Hlt}(f) = 1$.

Acceptance condition

$\mathcal{G}$ **accepts** (G, v) iff n visits an accepting feature vector at least once.

- Termination is *signaled by halting function*. There is no externally fixed number of rounds.

Reachability of a label p

Given $\mathcal{P}(P)$-labeled graph $G = (N, E, g)$.

Node Property: from n there is a path to some node labeled p.

$$g_0(n) = \begin{cases} 1 & p \in g(n) \\ 0 & \text{else} \end{cases} \qquad g_t(n) = \mathrm{trReLU}\Big(g_{t-1}(n) + \textstyle\sum_{(n,m)\in E} g_{t-1}(m)\Big)$$

with $\mathrm{trReLU}(x) = \min(\max(0, x), 1)$ and $f = 1$ accepting feature vector (scalar)

- After t rounds: $g_t(n) = 1$ iff some node labeled p is reachable from n in $\leq t$ steps.
- n is accepted iff it *ever* reaches 1: exactly reachability!
- Note: no constant-depth GNN (and no GML-formula) can do this; recurrence is essential.

Centre-point property

Node Property: There exists a $k \in \mathbb{N}$ such that every directed path from n reaches an end node in exactly k steps.

$$h_t(n) = (p_t(n), b_t(n)) \in \mathbb{R}^2,$$

Initialize

$$h_0(n) = (0, 0) \qquad \text{for all } n.$$

Aggregate out-neighbours E^+ by

$$\mathrm{Agg}\{\{h_t(m) \mid m \in E^+(n)\}\} = (d_t(n), s_t(n)),$$

where

$$d_t(n) = |E^+(n)|, \qquad s_t(n) = \sum_{m \in E^+(n)} b_t(m).$$

Centre-point property: combine

The aggregate gives

$$(d, s) = \left(|E^+(n)|, \sum_{m \in E^+(n)} b_t(m) \right).$$

Thus d is the number of out-neighbours, and $s = d$ means that all out-neighbours currently have value 1.

The combine map is

$$\mathrm{Comb}\big((p, b), (d, s)\big) = (1, B),$$

where

$$B = \begin{cases} 1 & \text{if } p = 0 \text{ and } d = 0, \\ 1 & \text{if } p = 1 \text{ and } d > 0 \text{ and } s = d, \\ 0 & \text{otherwise.} \end{cases}$$

Centre-point property: what the combine does

- The first round is a warm-up round:

$$b_1(n) = 1 \quad \Longleftrightarrow \quad n \text{ is an end node.}$$

- From then on, the same recurrent layer propagates the value backwards:

$$b_{t+1}(n) = 1$$

iff

$$n \text{ is not an end node and all out-neighbours } m \text{ satisfy } b_t(m) = 1.$$

- Therefore $b_{t+1}(n) = 1$ says: every directed path from n reaches an end node in exactly $t + 1$ steps.

The phase bit p separates the base case from the recursive case.

Directed to undirected

Encoding idea

Replace every directed edge $u \to v$ by a small undirected labelled gadget:

$$u \to v \qquad \rightsquigarrow \qquad u \; - \; e^{\text{out}}_{u,v} \; - \; e^{\text{in}}_{u,v} \; - \; v.$$

- The original nodes u, v keep their old labels.
- The two new nodes carry fresh labels

$$\text{out}, \qquad \text{in}.$$

- To follow a directed edge, move through the pattern

$$\text{out-node} \;\; \text{then} \;\; \text{in-node}.$$

Expressive power?

- As before, how to assess the expressive power of recurrent GNNs in the accepting state mode?
- Can we find logical characterization?

The answer is yes!... using extension of GML.

Syntax of ω-GML

Fix a finite set of propositions P

ω-**GML** extends graded modal logic with infinitary disjunctions:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \Diamond_{\geq k}\varphi$$

where $p \in P$, $k \geq 1$ and

$$\psi := \varphi \mid \bigvee_{i \in I} \varphi_i$$

and I ranges over possibly countable index sets.

ω-GML: semantics

Let $G = (N, E, g)$ be a $\mathcal{P}(P)$-labeled graph.

- $$G, n \models p \quad \iff \quad p \in g(n).$$
- Boolean connectives are interpreted as usual.
- $$G, n \models \Diamond_{\geq k} \varphi \text{ iff } \big|\{m \in E(n) \mid G, m \models \varphi\}\big| \geq k.$$
- $$G, n \models \bigvee_{i \in I} \varphi_i \text{ iff there exists an } i \text{ such that } G, n \models \varphi_i.$$

So: one (possibly infinite) disjunction over ordinary GML-formulae: no infinite conjunctions, no nesting of infinite disjunctions.

Reachability in ω-GML

Node property

From the current node n, there is a path to some node labelled p.

Define finite-depth GML formulae R_t by

$$R_0 \coloneqq p, \qquad R_{t+1} \coloneqq R_t \vee \Diamond R_t.$$

- R_t says: a p-node is reachable in at most t steps.
- Hence unbounded reachability is the ω-GML formula

$$\mathrm{Reach}_p \coloneqq \bigvee_{t \in \mathbb{N}_0} R_t.$$

- Equivalently,

$$\mathrm{Reach}_p \equiv \bigvee_{t \in \mathbb{N}_0} \Diamond^t p, \qquad \Diamond^0 p \coloneqq p, \quad \Diamond^{t+1} p \coloneqq \Diamond(\Diamond^t p).$$

Centre-point property in ω-GML

Node property

There exists a $k \in \mathbb{N}_0$ such that every directed path from n reaches an end node in exactly k steps.

Here $\Diamond$ range over outgoing neighbours.
Define finite-depth GML formulae C_t by

$$C_0 \coloneqq \Diamond_{=0} \top, \qquad C_{t+1} \coloneqq \Diamond C_t \wedge \neg \Diamond \neg C_t.$$

- C_0 says: n is already an end node.
- C_{t+1} says: n has at least one outgoing neighbour, and all outgoing neighbours satisfy C_t.
- Thus C_t says: every directed path from n reaches an end node in exactly t steps.

$$\text{Centre} \coloneqq \bigvee_{t \in \mathbb{N}_0} C_t.$$

Main theorem

Theorem

Recurrent GNNs in the accepting-state model and ω-GML have the same expressive power.

Proof strategy. Neither direction is proved directly. Both go through an intermediate *automaton* model that strips away the real numbers but keeps the message-passing structure:

$$\omega\text{-GML} \longleftrightarrow \text{CMPA} \longleftrightarrow \text{recurrent GNNs}$$

Why automata? Counting message passing automata

A recurrent GNN in the accepting-state model is just a message-passing machine whose "states" happen to be vectors in $\mathbb{R}^d$. Abstracting the state space gives:

Definition (CMPA)

A counting message passing automaton over X is $\mathcal{A} = (Q, \pi, \delta, F)$:

- Q: an at most **countable** set of states,
- $\pi \colon \mathcal{P}(X) \to Q, \quad \delta \colon Q \times \mathcal{M}(Q) \to Q, \quad F \subseteq Q.$

Computation and acceptance exactly as for GNNs: each node updates from its own state and the *multiset* of out-neighbour states; accept if an accepting state is ever visited.

- Same skeleton as a recurrent GNNs: only $\mathbb{R}^d$ is replaced by countable Q.

The key tool: r-types

- The r-type of a pointed graph (G, w) is its equivalence class under r-round *graded bisimulation* $\sim_r$: same labels at the points, and for r rounds, matching out-neighbours with exact counts.
- A characteristic formula for this type is a GML-formula $\vartheta^r_{G,w}$ of modal depth r such that, for every pointed graph (H, v),

$$H, v \models \vartheta^r_{G,w} \quad \iff \quad (H, v) \sim_r (G, w).$$

- Thus $\vartheta^r_{G,w}$ describes exactly what can be seen from w up to depth r.

- Each pointed graph has **exactly one** r-type for each r; there are countably many r-types overall.
- **Type decomposition:** every GML-formula of modal depth r is logically equivalent to a countable disjunction of formulae $\vartheta^r_{G,w}$. Hence so is every ω-GML-formula (disjunct by disjunct).

The counting type automaton

There is one "universal" CMPA whose run computes r-types:

Counting type automaton

- States Q = all characteristic formulae $\vartheta^r_{G,w}$ (all r, all (G, w); countable).
- π: maps a label set P to the 0-type $\bigwedge_{p\in P} p \wedge \bigwedge_{p\notin P} \neg p$.
- $\delta(\vartheta, N)$: from own r-type ϑ and the multiset N of the neighbours' r-types, assemble the $(r+1)$-type:

$$\delta(\vartheta, N) = \vartheta_0 \wedge \bigwedge_{\ell\geq 1} \{\Diamond_{=\ell}\, \sigma \mid N(\sigma) = \ell\} \wedge \Diamond_{=|N|} \top$$

  where ϑ_0 is the 0-type part (the labels) of ϑ.

Invariant: after round r, node w of G is in state $\vartheta^r_{G,w}$: the automaton *is* the type construction, run distributedly. Only the accepting set F remains to be chosen.

ω-GML $\Rightarrow$ CMPA

Given $\chi = \bigvee_{\varphi \in S} \varphi$:

- rewrite each φ (of depth r) as a disjunction of characteristic formulae $\vartheta^r_{G,w}$ (type decomposition);
- take the counting type automaton with $F = \{\text{all } \vartheta^r_{G,w} \text{ appearing in any disjunct}\}$.

Then $(G, w) \models \chi$ iff w visits an accepting state.

CMPA $\Rightarrow \omega$-GML

The crucial lemma:

Lemma (r-types determine runs)

$(G, w) \sim_r (H, v) \iff$ (G, w) and (H, v) are in the same state at every round $\leq r$, *in every* CMPA

(proof: induction on r; multisets of neighbour states $\leftrightarrow$ exact counts of neighbour r-types).

So an automaton's behaviour up to round r depends only on the r-type. Take

$$\Phi = \{\vartheta^r_{G,w} \mid \mathcal{A} \text{ accepts } (G, w) \text{ in round } r\}, \qquad \chi_{\mathcal{A}} := \bigvee_{\vartheta \in \Phi} \vartheta \ \in \ \omega\text{-GML}.$$

Closing the loop with recurrent GNNs

Recurrent GNNs $\Rightarrow$ ω-GML.

The lemma also holds verbatim for recurrent GNNs:
$(G, w) \sim_r (H, v) \Rightarrow$ same feature vector up to round r in every recurrent GNN (a GNN is just a message passer).

So again $\bigvee\{\vartheta^r_{G,w} \mid \mathcal{G} \text{ accepts } (G, w) \text{ in round } r\}$ works.

Closing the loop with recurrent GNNs

ω-GML $\Rightarrow$ Recurrent GNNs

Translate the formula to a counting type automaton, then simulate it by a GNN over using *natural numbers* as features:

- Each r-type $\leftrightarrow$ its minimal tree model $\leftrightarrow$ a binary string $\leftrightarrow$ an integer.
- AGG: decode the neighbour multiset's trees, hang them under a fresh root, re-encode.
- COM: fix the root's labels using the node's own code.
- F: the integers coding accepting r-types.

Recurrent GNNs evaluate ω-GML

For every ω-GML-formula χ, we can thus construct one recurrent GNN $\mathcal{N}_\chi$ such that, for all pointed graphs (G, w),

$$(G, w) \models \chi \quad \iff \quad \mathcal{N}_\chi \text{ accepts } w \text{ on } G.$$

What about distinguishing power

Proposition 8

Recurrent GNNs in the accepting state model coincide with non-recurrent GNNs when it comes to distinguishability.

Using logic.

$(G, m) \models \phi$ and $(H, n) \not\models \phi$ for ω-GML formula $\phi = \bigvee \psi_i$ iff exist i such that $(H, n) \not\models \psi_i$ and $(G, m) \models \psi_i$ for GML ψ.

Recall, for distinguishability $GML = GNN$ and recurrent GNNs $= \omega$-GNN.

Non-reachability is not expressible in ω-GML

Claim

No ω-GML formula φ satisfies, for every graph G and node m:

$$G, m \models \varphi \quad \iff \quad m \text{ does not reach } p.$$

Proof.

1. Suppose φ exists. Fix G, m with $m \not\to p$. Then $G, m \models \varphi$.

□

Non-reachability is not expressible in ω-GML

Claim

No ω-GML formula φ satisfies, for every graph G and node m:

$$G, m \models \varphi \quad \iff \quad m \text{ does not reach } p.$$

Proof.

1. Suppose φ exists. Fix G, m with $m \not\to p$. Then $G, m \models \varphi$.
2. φ is a (possibly infinite) disjunction $\bigvee_i \varphi_i$, so $G, m \models \varphi_i$ for some i; let k be the modal depth of φ_i.

□

Non-reachability is not expressible in ω-GML

Claim

No ω-GML formula φ satisfies, for every graph G and node m:

$$G, m \models \varphi \quad \iff \quad m \text{ does not reach } p.$$

Proof.

1. Suppose φ exists. Fix G, m with $m \not\to p$. Then $G, m \models \varphi$.
2. φ is a (possibly infinite) disjunction $\bigvee_i \varphi_i$, so $G, m \models \varphi_i$ for some i; let k be the modal depth of φ_i.
3. Build G', m' agreeing with m on its k-neighborhood, but where p is reachable from m' in exactly $k+1$ steps.

□

Non-reachability is not expressible in ω-GML

Claim

No ω-GML formula φ satisfies, for every graph G and node m:

$$G, m \models \varphi \quad \iff \quad m \text{ does not reach } p.$$

Proof.

1. Suppose φ exists. Fix G, m with $m \not\to p$. Then $G, m \models \varphi$.
2. φ is a (possibly infinite) disjunction $\bigvee_i \varphi_i$, so $G, m \models \varphi_i$ for some i; let k be the modal depth of φ_i.
3. Build G', m' agreeing with m on its k-neighborhood, but where p is reachable from m' in exactly $k+1$ steps.
4. Depth-k formulas cannot distinguish m from m', so $G', m' \models \varphi_i$, hence $G', m' \models \varphi$ — but m' reaches p. Contradiction.

□

Conclusion: Accepting state model

- Non-reachability and any other property not in ω-GML cannot be compiled into the recurrent network model.
- Everything in ω-GML can be compiled, however.

Good and bad: accepting-state model

Good

- Very simple semantics:
- Natural for eventual or recursive properties.
- Captures reachability-like computations directly:

$$p \vee \Diamond p \vee \Diamond^2 p \vee \cdots$$

- Clean logical characterisation:

accepting-state RGNNs $\equiv \omega$-GML.

Bad

- Acceptance is existential in time: accept if an accepting state is reached at some round.
- Negative instances do not produce a final rejecting state; they simply never accept.
- Thus the model is closer to a semi-decision procedure than to a terminating classifier.
- Complements need not be expressible: reachability is expressible, but non-reachability is not.

Part 5 (b): Basics of halting recurrent GNNs

- What is covered next:
  - Introduction of halting-classifier-based recurrent GNNs.
  - We prove that on connected graph these GNNs have the same distinguishing power as fixed-depth GNNs.
  - The model can express reachability (shown here) and non-reachability (shown later).
  - Most likely, the model cannot express centre-point property.
- Reference:
  - Jeroen Bollen, Jan Van den Bussche, Stijn Vansummeren, Jonni Virtema. Halting Recurrent GNNs and the Graded μ-Calculus. KR 2025.

Halting model: Intuition

- AC-GNN with no limit on the number of layers.
- Each node internally indicates when their feature vector is ready to be read.
- Global read-out is used only to check when all nodes are ready to halt.
  - This indicates the final layer of the GNN!

Halting model: Formally

Let X be a finite input label set.

- A halting-classifier-based recurrent GNN is a tuple

$$\mathcal{N} = (\mathsf{In}, L, \mathsf{Hlt})$$

- The map

$$\mathsf{In} : X \to \mathbb{R}^d$$

initializes node features of an input graph $\mathcal{G}(X)$.

- The layer L is an aggregate-combine layer

$$L : \mathcal{G}[\mathbb{R}^d] \to \mathcal{G}[\mathbb{R}^d].$$

- The function

$$\mathsf{Hlt} : \mathbb{R}^d \to \mathbb{B}$$

decides whether a node is ready to halt.

Runs in the halting model

Let $G = (N, E, g)$ be an X-labeled graph.

- The initial graph is
$$H_0 \coloneqq \mathsf{In}^{\uparrow}(G).$$
- The recurrent layer is applied repeatedly:
$$H_{i+1} \coloneqq L(H_i).$$
- A run is complete at time k if every node halts:
$$\mathsf{Hlt}(h_k(n)) = 1 \qquad \text{for all } n \in N,$$
and k is the smallest number where this holds.
- The output of graph is then $\mathcal{N}(G) = H_k$.

Runs in the halting model

Let $G = (N, E, g)$ be an X-labeled graph.

- The initial graph is
$$H_0 := \mathsf{In}^{\uparrow}(G).$$
- The recurrent layer is applied repeatedly:
$$H_{i+1} := L(H_i).$$
- A run is complete at time k if every node halts:
$$\mathsf{Hlt}(h_k(n)) = 1 \qquad \text{for all } n \in N,$$
and k is the smallest number where this holds.
- The output of graph is then $\mathcal{N}(G) = H_k$.
- Difference to the accepting state model:
  - Halting relies on a global read-out. Layer-uniform output-layer is H_k.

Halting model

- A recurrent GNN is halting if every finite input graph has a complete run.
- Let $\mathcal{N}$ be a halting GNN with with Agg-Comb layer L.
  - For every input graph G there exists k such that $\mathcal{N}(G) = H_k$.
  - If $\mathcal{N}_k$ is the (non-recursive) k-layer GNN whose layers compute L, then $\mathcal{N}_k(G) = H_k$

Halting model

- A recurrent GNN is halting if every finite input graph has a complete run.
- Let $\mathcal{N}$ be a halting GNN with with Agg-Comb layer L.
  - For every input graph G there exists k such that $\mathcal{N}(G) = H_k$.
  - If $\mathcal{N}_k$ is the (non-recursive) k-layer GNN whose layers compute L, then $\mathcal{N}_k(G) = H_k$
- Caviats:
  - Halting function does not indicate when computation halts locally.
  - Output layer of the halting model is the first layer where all nodes indicate halting.
  - In general, a node may alternate between being ready or not to halt.

Halting model

- A recurrent GNN is halting if every finite input graph has a complete run.
- Let $\mathcal{N}$ be a halting GNN with with Agg-Comb layer L.
  - For every input graph G there exists k such that $\mathcal{N}(G) = H_k$.
  - If $\mathcal{N}_k$ is the (non-recursive) k-layer GNN whose layers compute L, then $\mathcal{N}_k(G) = H_k$
- Caviats:
  - Halting function does not indicate when computation halts locally.
  - Output layer of the halting model is the first layer where all nodes indicate halting.
  - In general, a node may alternate between being ready or not to halt.
  - Recurrent GNNs may define partial node-feature maps. All GNNs are not Halting!
  - Solution: Only consider GNNs that are halting.

Example: Reachability in a given bound

- The halting mechanism allows simulation of various counters.
  (HALT when counter reaches 0).
- Consider the following node property for pointed graphs (G, n):
  Does there exists path from n to a node satisfying p in at most $\deg(n)$ many steps.

Example: Reachability in a given bound

- The halting mechanism allows simulation of various counters.
  (HALT when counter reaches 0).
- Consider the following node property for pointed graphs (G, n):
  Does there exists path from n to a node satisfying p in at most $\deg(n)$ many steps.
- It is easy to design a halting GNN that
  - sets the value of one feature to $\deg(n)$,
  - in each successive layer, decreases this counter by one,
  - HALTS when this feature reaches 0,
  - in a second feature, the GNN computes the truth value of a GML formula $\bigvee_{j \leq i} \Diamond^j p$, for increasing values of i.

Example: Reachability in a given bound

- The halting mechanism allows simulation of various counters.
  (HALT when counter reaches 0).
- Consider the following node property for pointed graphs (G, n):
  Does there exists path from n to a node satisfying p in at most $\deg(n)$ many steps.
- It is easy to design a halting GNN that
  - sets the value of one feature to $\deg(n)$,
  - in each successive layer, decreases this counter by one,
  - HALTS when this feature reaches 0,
  - in a second feature, the GNN computes the truth value of a GML formula $\bigvee_{j \leq i} \Diamond^j p$, for increasing values of i.
- The implementation of L is the same as in the accepting state model.

Example: Reachability in a given bound

- The halting mechanism allows simulation of various counters.
  (HALT when counter reaches 0).
- Consider the following node property for pointed graphs (G, n):
  Does there exists path from n to a node satisfying p in at most $\deg(n)$ many steps.
- It is easy to design a halting GNN that
  - sets the value of one feature to $\deg(n)$,
  - in each successive layer, decreases this counter by one,
  - HALTS when this feature reaches 0,
  - in a second feature, the GNN computes the truth value of a GML formula $\bigvee_{j \leq i} \Diamond^j p$, for increasing values of i.
- The implementation of L is the same as in the accepting state model.
- When this GNN halts, each node n has computed the truth value of the formula

$$\bigvee_{j \leq \deg n} \Diamond^j p.$$

Example: General Reachability

- It is not immediately clear how to design a halting GNN for checking whether there exist a path to a node satisfying p.

Example: General Reachability

- It is not immediately clear how to design a halting GNN for checking whether there exist a path to a node satisfying p.
- Easy part: Compute the truth value of $\bigvee_{j \leq i} \Diamond^j p$ for increasing values of i, and HALT and return 1 if p is reached. As in the accepting state model.
- Tricky part: How does the GNN know to HALT, if there is no p reachable?

Example: General Reachability

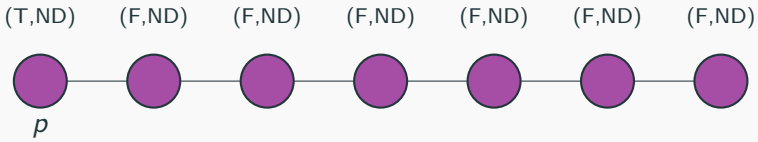
- Easy part: Compute the truth value of $\bigvee_{j \leq i} \Diamond^j p$ for increasing values of i, and HALT and return 1 if p is reached. As in the accepting state model.
- Tricky part: How does the GNN know to HALT, if there is no p reachable?
- The GNN will use two features
  - one to compute the truth value of $\bigvee_{j \leq i} \Diamond^j p$ for increasing values of i, and
  - another to compute the truth value of $\bigvee_{j \leq i-1} \Diamond^j p$ for increasing values of i.
  - HALT will output 1 iff the value of these features are equal (and $i \geq 1$).

Example: General Reachability

- Easy part: Compute the truth value of $\bigvee_{j \leq i} \Diamond^j p$ for increasing values of i, and HALT and return 1 if p is reached. As in the accepting state model.
- Tricky part: How does the GNN know to HALT, if there is no p reachable?
- The GNN will use two features
  - one to compute the truth value of $\bigvee_{j \leq i} \Diamond^j p$ for increasing values of i, and
  - another to compute the truth value of $\bigvee_{j \leq i-1} \Diamond^j p$ for increasing values of i.
  - HALT will output 1 iff the value of these features are equal (and $i \geq 1$).
- After halting, each n has computed the value of $\bigvee_{j \leq i} \Diamond^j p$, for some value of i.
- Why does this correctly indicate whether there is a path from n to a p node?

Illustration of the GNN computation

- Computation of reachability to p using the method of the previous page.
- After i layers, the figure shows the truth value of formulas $(\bigvee_{j \leq i} \Diamond^j p, \bigvee_{j \leq i-1} \Diamond^j p)$
- A node is green, if that node is ready to halt.

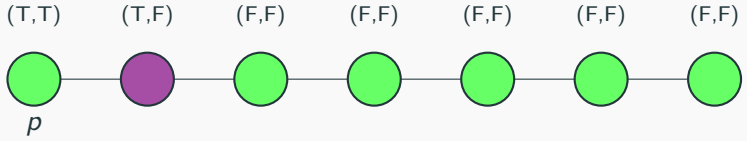


$\varphi_0 = p$

φ_{-1} undefined

Illustration of the GNN computation

- Computation of reachability to p using the method of the previous page.
- After i layers, the figure shows the truth value of formulas $(\bigvee_{j \leq i} \Diamond^j p, \bigvee_{j \leq i-1} \Diamond^j p)$
- A node is green, if that node is ready to halt.

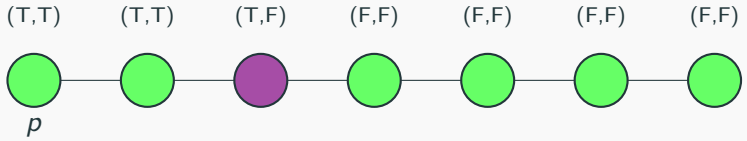


$$\varphi_1 = p \vee \Diamond p$$

$$\varphi_0 = p$$

Illustration of the GNN computation

- Computation of reachability to p using the method of the previous page.
- After i layers, the figure shows the truth value of formulas $(\bigvee_{j \leq i} \Diamond^j p, \bigvee_{j \leq i-1} \Diamond^j p)$
- A node is green, if that node is ready to halt.

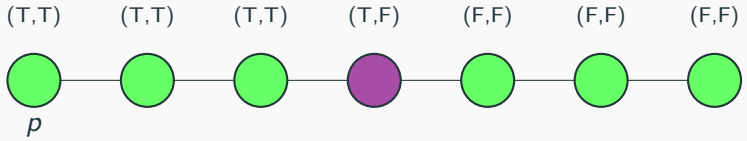


$$\varphi_2 = p \vee \Diamond(p \vee \Diamond p)$$

$$\varphi_1 = p \vee \Diamond p$$

Illustration of the GNN computation

- Computation of reachability to p using the method of the previous page.
- After i layers, the figure shows the truth value of formulas $(\bigvee_{j\leq i} \Diamond^j p, \bigvee_{j\leq i-1} \Diamond^j p)$
- A node is green, if that node is ready to halt.

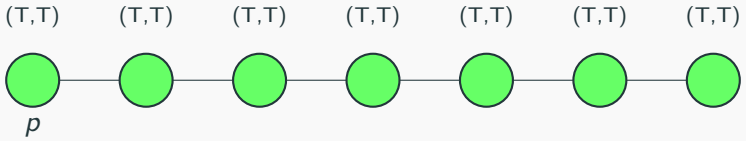


$$\varphi_3 = p \vee \Diamond\varphi_2$$

$$\varphi_2 = p \vee \Diamond\varphi_1$$

Illustration of the GNN computation

- Computation of reachability to p using the method of the previous page.
- After i layers, the figure shows the truth value of formulas $(\bigvee_{j \leq i} \Diamond^j p, \bigvee_{j \leq i-1} \Diamond^j p)$
- A node is green, if that node is ready to halt.



$$\varphi_7 = p \vee \Diamond \varphi_6$$

$$\varphi_6 = p \vee \Diamond \varphi_5$$

Every node agree on the truth value of φ_7 and φ_6

Hence the GNN halts

Each node outputs 1, the truth value of φ_6

Proposition 9

Let (G, n) and (H, m) be two pointed X-labeled graphs. Let χ_∞^G denote the stable colouring of colour refinement of G.

1. *For every halting GNN $\mathcal{N}$ with node features g, we have*

$$\chi_\infty^G(n) = \chi_\infty^H(m) \text{ and } \operatorname{set}(\operatorname{hist}_{\chi_\infty^G}) = \operatorname{set}(\operatorname{hist}_{\chi_\infty^H}) \implies g_{h_G}^G(n) = g_{h_H}^H(m),$$

   where h_H and h_G are the halting layers of $\mathcal{N}(G)$ and $\mathcal{N}(H)$. Moreover, $h_G = h_H$.

2. *Conversely, if colour refinement distinguishes the two pointed graphs, that is,*

$$\chi_\infty^G(n) \neq \chi_\infty^H(m),$$

   then there exists a halting GNN such that

$$g_k^G(n) \neq g_k^H(m).$$

Proof of Proposition 9: First direction

Proof.

- Assume $\chi_\infty^G(n) = \chi_\infty^H(m)$. In particular $\chi_l^G(n) = \chi_l^H(m)$, for every $l \in \mathbb{N}$.

Proof of Proposition 9: First direction

Proof.

- Assume $\chi_\infty^G(n) = \chi_\infty^H(m)$. In particular $\chi_l^G(n) = \chi_l^H(m)$, for every $l \in \mathbb{N}$.
- For any fixed number of layers l, the Halting GNN computes graph transformations identical to an l-layer GNN.
  - $g_t^G(n) = g_t^H(m)$ follows from Proposition 1 (this proposition for non-rec GNNs).

Proof of Proposition 9: First direction

Proof.

- Assume $\chi^G_\infty(n) = \chi^H_\infty(m)$. In particular $\chi^G_l(n) = \chi^H_l(m)$, for every $l \in \mathbb{N}$.
- For any fixed number of layers l, the Halting GNN computes graph transformations identical to an l-layer GNN.
  - $g^G_t(n) = g^H_t(m)$ follows from Proposition 1 (this proposition for non-rec GNNs).
- Since $\mathrm{set}(\mathrm{hist}_{\chi^G_\infty}) = \mathrm{set}(\mathrm{hist}_{\chi^H_\infty})$, in particular $\mathrm{set}(\mathrm{hist}_{\chi^G_l}) = \mathrm{set}(\mathrm{hist}_{\chi^H_l})$ for every l. Hence H and G halt at the same layer.

□

Proof of Proposition 9: Second direction.

- For any two pointed graphs $\chi^G_\infty(n) \neq \chi^H_\infty(m)$ iff $\chi^G_l(n) \neq \chi^H_l(m)$, for some $l \in \mathbb{N}$.

Proof of Proposition 9: Second direction.

- For any two pointed graphs $\chi^G_\infty(n) \neq \chi^H_\infty(m)$ iff $\chi^G_l(n) \neq \chi^H_l(m)$, for some $l \in \mathbb{N}$.
- We will show next that the distinguishing power of halting recurrent GNNs is at least that of non-rec GNNs.
- The result then follows from Proposition 1 (this proposition for non-recurrent GNNs).

Distinguishing power: fixed-depth versus recurrent

Proposition 10

- *If two finite pointed graphs can be distinguished by a fixed-depth GNN they can be distinguished by a halting recurrent GNN.*
- *If two connected finite pointed graphs can be distinguished by a halting recurrent GNN then they can be distinguished by a fixed-depth GNN.*

- Thus recurrence does not add distinguishing power for connected graphs.
- This is a non-uniform statement.
- The fixed-depth GNN obtained from a recurrent GNN may depend on the two pointed graphs being separated.

Proof: fixed-depth implies recurrent

- Suppose a fixed-depth GNN $\mathcal{N}$ with ℓ layers distinguishes (G, n) and (H, m).
- We have proven that $\mathcal{N}$ can be assumed to be homogeneous.
- A recurrent GNN can simulate this by storing a counter from 0 to ℓ.
- It applies the unique layer of $\mathcal{N}$ at each step.
- It halts after exactly ℓ rounds.
- Therefore every fixed-depth distinction can also be made by a recurrent GNN.

Proof: recurrent implies fixed-depth

Proof by contraposition:

- Two connected graphs cannot be distinguished by a fixed-depth GNN
  $\Rightarrow$ the graphs cannot be distinguished by a halting recurrent GNN.

1. Let (G, n) and (H, m) be two connected finite pointed graphs that cannot be distinguished by any fixed-depth GNN.
2. By Proposition 1, (G, n) and (H, m) cannot be separated by any finite-round CR, i.e., $\chi_t^G(n) = \chi_t^H(m)$, for each $t \in \mathbb{N}$. Hence $\chi_\infty^G(n) = \chi_\infty^H(m)$.
3. For connected graphs this implies that $\mathrm{set}(\mathrm{hist}_{\chi_\infty^G}) = \mathrm{set}(\mathrm{hist}_{\chi_\infty^H})$.
4. Proposition 9 (claim 1) then implies that no recurrent GNN can separate (G, n) and (H, m).

Logic...strikes back

- With respect to connected graphs, the distinguishing power of halting RGNNs coincide with fixed-layer GNNs and GML.
- Uniform expressivity of halting RGNNs goes past GML (e.g., reachability).
  - Is there a logical counterpart for halting RGNNs?

Part 5 (c): Halting recurrent GNNs and modal μ-calculus

From GML to graded modal μ-calculus

- GML describes bounded-depth local properties.
- A formula of modal depth r can only look r steps away.
- To express iterative properties such as reachability, we proceed in two ways:
  - add infinitary disjunctions, or
  - add fixpoint operators.
- This gives us ω-GML and graded modal μ-calculus, denoted μGML, respectively.

Syntax of μGML

Fix a finite set of propositions P and a set of variables Var.

- Formulae of graded modal μ-calculus are generated by

$$\varphi ::= p \mid \neg p \mid X \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \Diamond_{\geq k} \varphi \mid \mu X.\varphi \mid \nu X.\varphi,$$

  where $p \in P$, $X \in \mathrm{Var}$, and $k \geq 1$.

- The modality $\Diamond_{\geq k} \varphi$ means:

$$\text{at least } k \text{ neighbours satisfy } \varphi.$$

- The formula $\mu X.\varphi$ denotes a least fixpoint.
- The formula $\nu X.\varphi$ denotes a greatest fixpoint ($\nu X.\varphi \equiv \neg \mu X.\neg \varphi$).
- Variables must occur positively in fixpoint bodies, so that the induced operator is monotone.

Intuition for fixed points

- We write $\varphi[\psi/X]$ for the formula obtained from φ by substituting X with ψ.
- Let $\varphi := \Diamond X \vee p$,
- Define $\varphi_0 := \varphi[\bot/X] = \Diamond\bot \vee p$, which is equivalent to p.
- Define $\varphi_1 := \varphi[\varphi_0/X] = \Diamond(\Diamond\bot \vee p) \vee p$, which is equivalent to $\Diamond p \vee p$.
- Define $\varphi_2 := \varphi[\varphi_1/X] = \Diamond(\Diamond(\Diamond\bot \vee p) \vee p) \vee p$, is equivalent to $\Diamond\Diamond p \vee \Diamond p \vee p$.

Intuition for fixed points

- We write $\varphi[\psi/X]$ for the formula obtained from φ by substituting X with ψ.
- Let $\varphi := \Diamond X \vee p$,
- Define $\varphi_0 := \varphi[\bot/X] = \Diamond\bot \vee p$, which is equivalent to p.
- Define $\varphi_1 := \varphi[\varphi_0/X] = \Diamond(\Diamond\bot \vee p) \vee p$, which is equivalent to $\Diamond p \vee p$.
- Define $\varphi_2 := \varphi[\varphi_1/X] = \Diamond(\Diamond(\Diamond\bot \vee p) \vee p) \vee p$, is equivalent to $\Diamond\Diamond p \vee \Diamond p \vee p$.
- Define $\varphi_{i+1} := \varphi[\varphi_i/X]$ is equivalent to $\bigvee_{j\leq i} \Diamond^i p$.
- Hence φ_i is ϕ unravelled recursively within itself i times.

Intuition for fixed points

- We write $\varphi[\psi/X]$ for the formula obtained from φ by substituting X with ψ.
- Let $\varphi := \Diamond X \vee p$,
- Define $\varphi_0 := \varphi[\bot/X] = \Diamond\bot \vee p$, which is equivalent to p.
- Define $\varphi_1 := \varphi[\varphi_0/X] = \Diamond(\Diamond\bot \vee p) \vee p$, which is equivalent to $\Diamond p \vee p$.
- Define $\varphi_2 := \varphi[\varphi_1/X] = \Diamond(\Diamond(\Diamond\bot \vee p) \vee p) \vee p$, is equivalent to $\Diamond\Diamond p \vee \Diamond p \vee p$.
- Define $\varphi_{i+1} := \varphi[\varphi_i/X]$ is equivalent to $\bigvee_{j\leq i} \Diamond^j p$.
- Hence φ_i is ϕ unravelled recursively within itself i times.
- $(G, m) \models \mu X.\varphi(X)$ iff $(G, m) \models \varphi_i$, for some i.
- Intuition: $\mu X.\varphi(X)$ is true if the recursive procedure defined by φ holds after finite number of recursion.

Semantics: environments

Let $G = (N, E, g)$ be a $\mathcal{P}(P)$-labeled graph.

- To interpret variables, we use an environment

$$\eta : \mathsf{Var} \to \mathcal{P}(N).$$

- Thus $\eta(X)$ is the set of nodes currently assigned to the variable X.
- For variables,

$$G, n \models_\eta X \quad \iff \quad n \in \eta(X).$$

- Atomic propositions and graded modalities are interpreted as before:

$$G, n \models_\eta \Diamond_{\geq k} \varphi$$

iff

$$|\{m \in E(n) \mid G, m \models_\eta \varphi\}| \geq k.$$

Least fixpoints

Consider a formula $\mu X.\varphi(X)$.

- The body $\varphi(X)$ defines a monotone operator

$$F : \mathcal{P}(N) \to \mathcal{P}(N),$$

  where

$$F(S) := \{n \in N \mid G, n \models_{\eta[X \mapsto S]} \varphi\}.$$

- The formula $\mu X.\varphi(X)$ denotes the least fixpoint of F:

$$G, n \models_\eta \mu X.\varphi \quad \iff \quad n \in \mathrm{lfp}(F).$$

- On finite graphs, the least fixpoint is obtained by iteration:

$$S_0 := \emptyset, \qquad S_{i+1} := F(S_i).$$

Example: reachability

Fix an atomic proposition p.

- The formula
$$\mathrm{Reach}_p := \mu X.\big(p \vee \Diamond X\big)$$
says that a node can reach a p-node.
- The least-fixpoint iteration is
$$R_0 := \emptyset,$$
$$R_{i+1} := \{n \mid p \in g(n)\} \cup \{n \mid E(n) \cap R_i \neq \emptyset\}.$$
- At the fixpoint, R_i is exactly the set of nodes from which some p-node is reachable.

μGML formulae as node classifiers

Let φ be a closed μGML formula.

- The formula φ defines a node classifier

$$C_\varphi : \mathcal{G}[\mathcal{P}(P)] \to \mathcal{G}[\mathbb{B}].$$

- For each graph $G = (N, E, g)$,

$$C_\varphi(G) = (N, E, c_G^\varphi),$$

where

$$c_G^\varphi(n) = 1 \quad \iff \quad G, n \models \varphi.$$

- Thus μGML extends GML from bounded-depth classifiers to iterative classifiers.

Simple recurrent GNNs

- We focus on simple recurrent GNNs.
- The aggregation is sum aggregation:

$$\mathrm{Agg}(M) = \sum_{x \in \mathrm{supp}(M)} M(x) \cdot x.$$

- The combination function has the form

$$\mathrm{Comb}(x, y) = f(x \mid y),$$

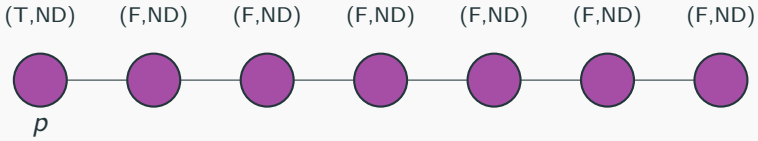
  where f is a ReLU feedforward neural network.
- The halting and output functions are simple threshold tests on coordinates.
- Thus the recurrence comes from repeatedly applying the same ordinary message-passing layer.

Evaluating μGML with GNNs

- Typical approach: a feature vector is used to recursively compute the truth values of subformulas of φ.
  - The initialisation map uses one-hot encoding for proposition symbols.
  - GNN layers are used to compute recursively truth values of composite formulae.
- Challenge: If we compute fixed points recursively, the number of subformulas is not fixed.
- Solution: Develop approximation semantics for μ-calculus.
- Recall how reachability was expressed!

Illustration of the GNN computation

- Computation of reachability to p using the method of the previous page.
- After i layers, the figure shows the truth value of formulas $(\bigvee_{j \leq i} \Diamond^j p, \bigvee_{j \leq i-1} \Diamond^j p)$
- A node is green, if that node is ready to halt.

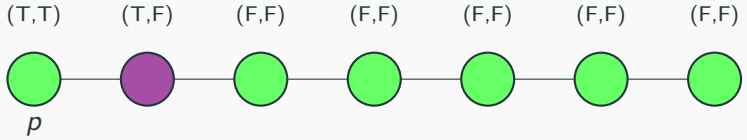


$\varphi_0 = p$

φ_{-1} undefined

Illustration of the GNN computation

- Computation of reachability to p using the method of the previous page.
- After i layers, the figure shows the truth value of formulas $(\bigvee_{j \leq i} \Diamond^j p, \bigvee_{j \leq i-1} \Diamond^j p)$
- A node is green, if that node is ready to halt.

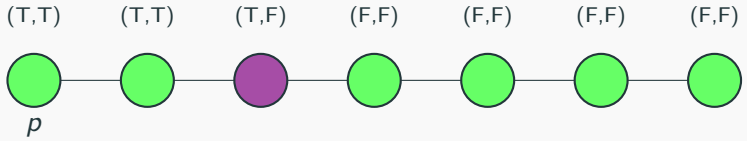


$$\varphi_1 = p \vee \Diamond p$$

$$\varphi_0 = p$$

Illustration of the GNN computation

- Computation of reachability to p using the method of the previous page.
- After i layers, the figure shows the truth value of formulas $(\bigvee_{j \leq i} \Diamond^j p, \bigvee_{j \leq i-1} \Diamond^j p)$
- A node is green, if that node is ready to halt.

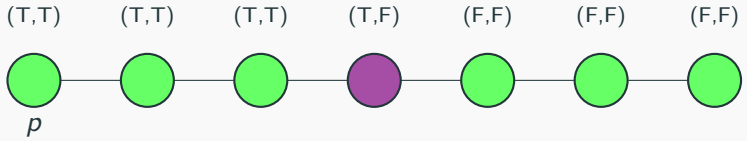


$$\varphi_2 = p \vee \Diamond(p \vee \Diamond p)$$

$$\varphi_1 = p \vee \Diamond p$$

Illustration of the GNN computation

- Computation of reachability to p using the method of the previous page.
- After i layers, the figure shows the truth value of formulas $(\bigvee_{j\leq i} \Diamond^j p, \bigvee_{j\leq i-1} \Diamond^j p)$
- A node is green, if that node is ready to halt.

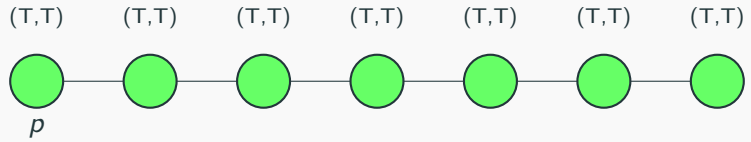


$$\varphi_3 = p \vee \Diamond\varphi_2$$

$$\varphi_2 = p \vee \Diamond\varphi_1$$

Illustration of the GNN computation

- Computation of reachability to p using the method of the previous page.
- After i layers, the figure shows the truth value of formulas $(\bigvee_{j \leq i} \Diamond^j p, \bigvee_{j \leq i-1} \Diamond^j p)$
- A node is green, if that node is ready to halt.



$$\varphi_7 = p \vee \Diamond \varphi_6$$

$$\varphi_6 = p \vee \Diamond \varphi_5$$

Every node agree on the truth value of φ_7 and φ_6

Hence the GNN halts

Each node outputs 1, the truth value of φ_6

Approximate semantics of μ-calculus

- Recall: $G, m \models \mu X.\varphi(X)$ iff $(G, m) \models \varphi_i$, for some i, where φ_i is the ith unravelling of $\varphi(X)$.
- Monotonicity: If $G, m \models \varphi_i$ then $(G, m) \models \varphi_{i+1}$.
- Idea: Each node computes the truth values of $(\varphi_{i+1}, \varphi_i)$.
  - When all nodes agree on those values, their truth value is $\mu X.\varphi(X)$.
- Complex formula $G, m \models \mu X \mu Y.\varphi(X, Y)$ uses more complex counters $(\varphi_{(i,j)}, \varphi_{(i,j+1)})$.
  - Here $(3, 2)$ that we are in the third unravelling of X inside of which Y is unravelled twice.

Uniform expressivity of recurrent GNNs

- Node properties are invariant under (total) surjective g-bisimulations.
  If $(G, n) \in P$ and $f : G \to G'$ is a g-bisimulation, total and surjective function, then $(G', f(n)) \in P$
  - If two points are g-bisimilar, after any finite iterations, they are labelled with the same feature vector.
  - Totality and surjectivity imply that the two computations halt at the same iteration.
- On connected graphs, g-bisimilarity and total surjective g-bimilarity coincide.
- Every μGML-definable node property can be decided with a recurrent GNN.

Part 6: Further directions

- What have we covered:
  - non-recurrent and two models of recurrent GNNs;
  - connection of graded model logic and extensions thereof.
- What further directions could be considered
  - Going beyond boolean logic by considering real-valued logics.
  - More fine-grained analysis relating impact of specific choices of aggregation functions, non-linear activation functions to expressive power.
  - Study impact of computational power of Comb and Agg. For example, restrict those functions to come from some function complexity class (non-recurrent vs recurrent arithmetic circuits).
  - Taking training process into account. If it can be expressed, can it also be learned? That is, can one construct training data when used to train GNNs expresses the desired property?