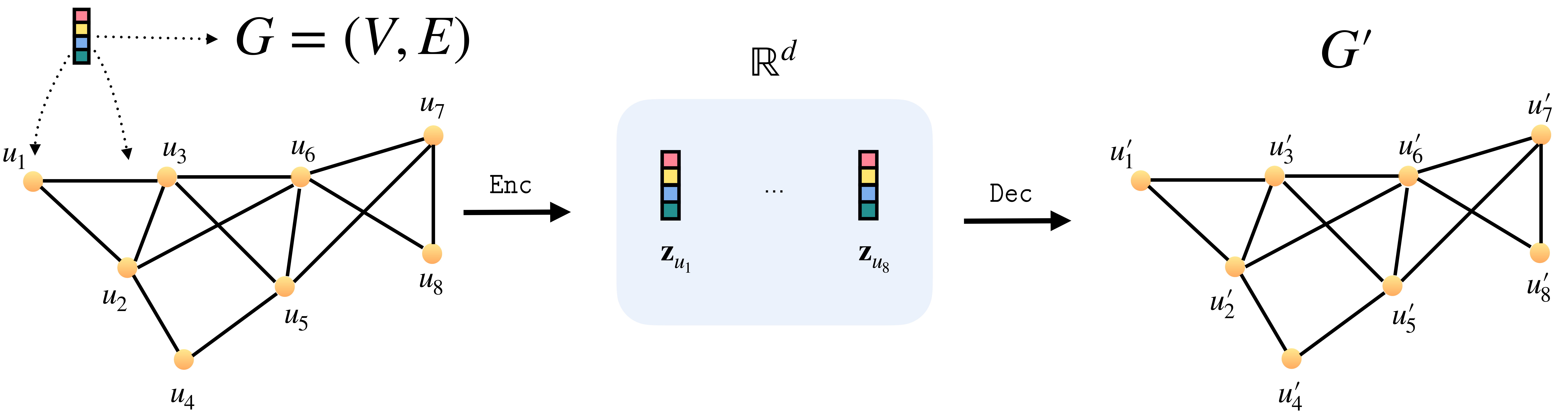
Lecture 5: Expressive Power of Message Passing Neural Networks

Relational Learning

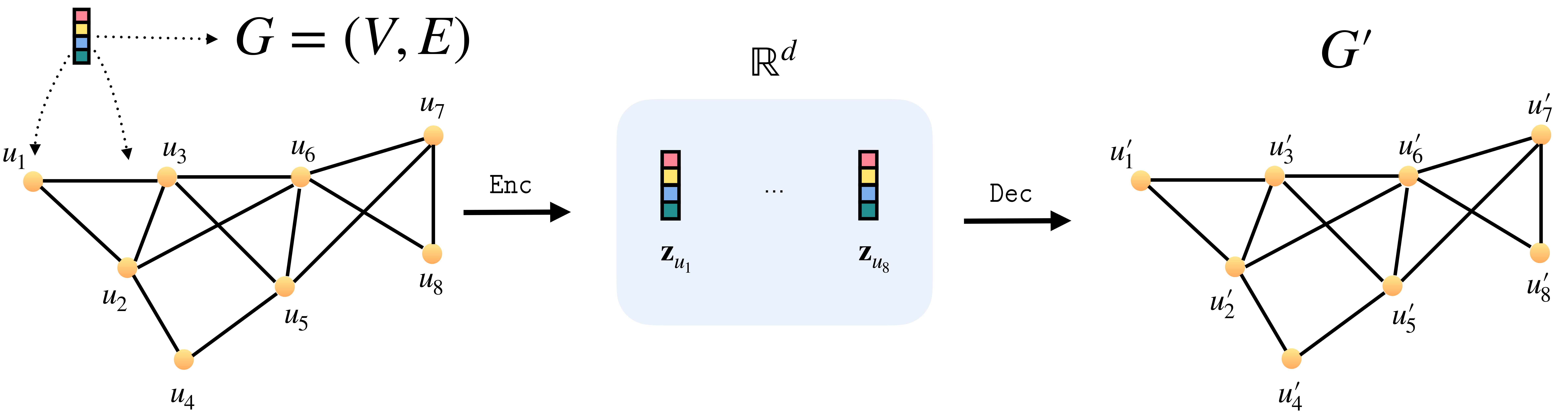
Graph Representation Learning



Graph representation learning with strong relational inductive bias

$$\mathbf{h}_u^{(t)} = \sigma\left(\mathbf{W}_{self}^{(t)}\mathbf{h}_u^{(t-1)} + \mathbf{W}_{neigh}^{(t)} \sum_{v \in N(u)} \mathbf{h}_v^{(t-1)}\right)$$

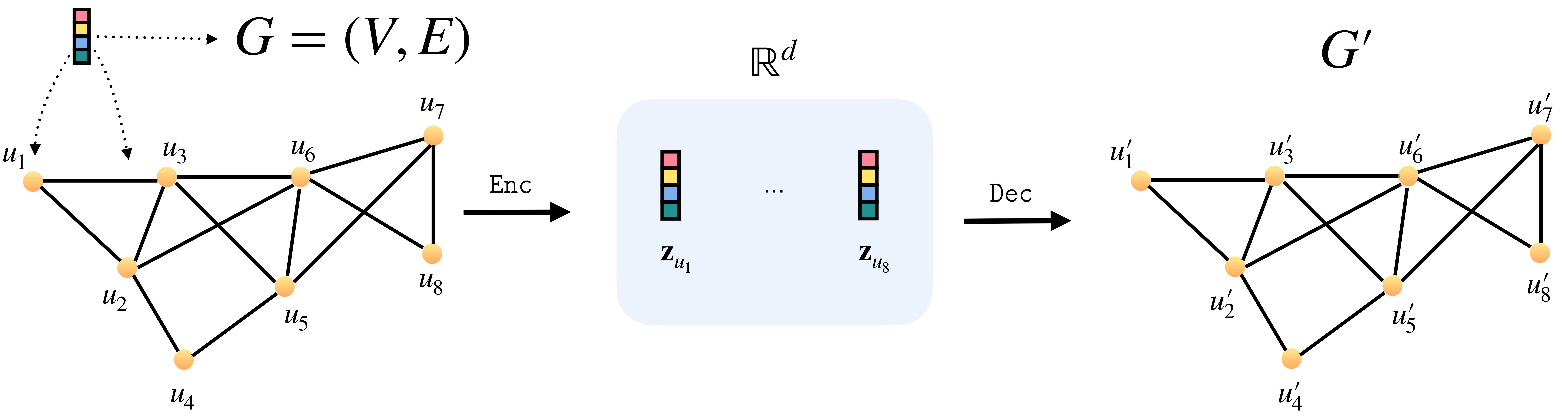
Graph Representation Learning



Learned parameters are independent of graph size

$$\mathbf{h}_u^{(t)} = \sigma\left(\mathbf{W}_{self}^{(t)}\mathbf{h}_u^{(t-1)} + \mathbf{W}_{neigh}^{(t)} \sum_{v \in N(u)} \mathbf{h}_v^{(t-1)}\right)$$

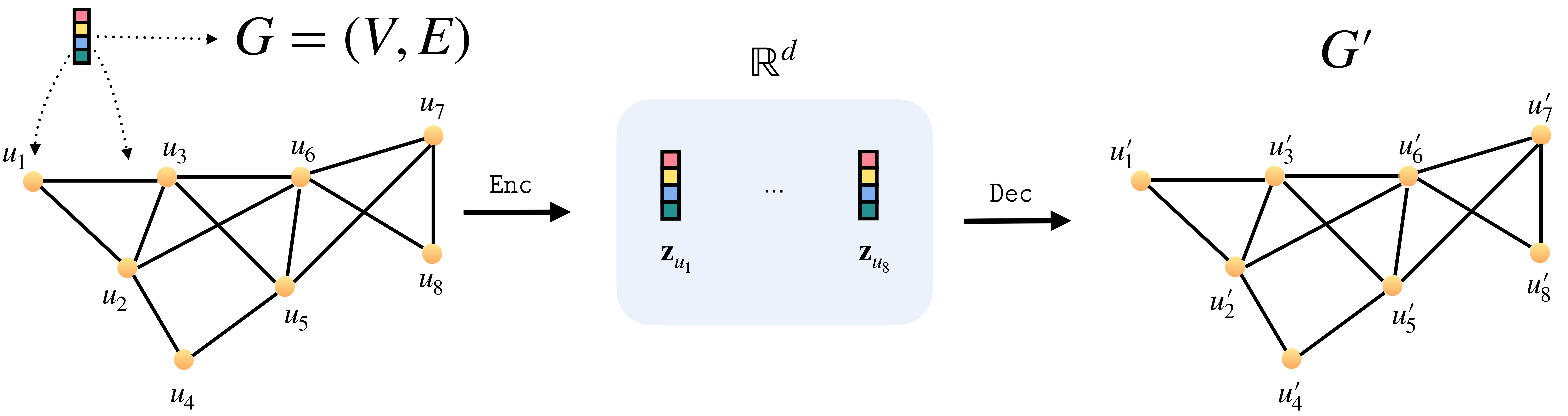
Graph Representation Learning



Applies to variable-size graphs

$$\mathbf{h}_u^{(t)} = \sigma\left(\mathbf{W}_{self}^{(t)}\mathbf{h}_u^{(t-1)} + \mathbf{W}_{neigh}^{(t)} \sum_{v \in N(u)} \mathbf{h}_v^{(t-1)}\right)$$

Graph Representation Learning



What is the expressive power?

$$\mathbf{h}_u^{(t)} = \sigma\left(\mathbf{W}_{self}^{(t)}\mathbf{h}_u^{(t-1)} + \mathbf{W}_{neigh}^{(t)} \sum_{v \in N(u)} \mathbf{h}_v^{(t-1)}\right)$$

Overview

- A journey into model representation capacity
- Graph isomorphism and color refinement
- Expressive power of message passing neural networks
- The logic of graphs
- Logical characterization of message passing neural networks
- Summary

A Journey into Model Representation Capacity

Model Representation Capacity

Expressive power: Capacity of a model (e.g., neural network) to approximate functions.

Universal approximation: MLPs can approximate any continuous function on a compact domain, i.e., for any such function, there is a parameter configuration for an MLP, corresponding to an approximation of the function (Cybenko, 1989; Funahashi, 1989; Hornik et al., 1989).

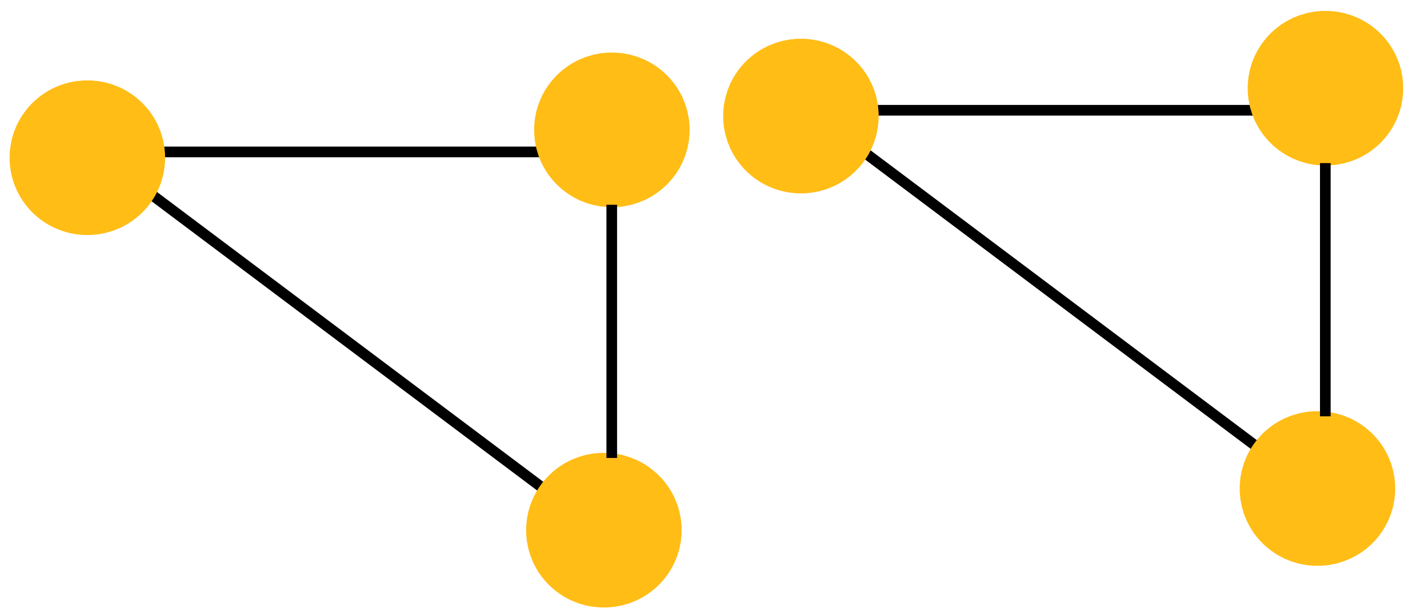
Graphs: One way of characterizing the expressive power would be through graph distinguishability. Learn graph embeddings $\mathbf{z}_G$, $\mathbf{z}_H$ for graphs G and H:

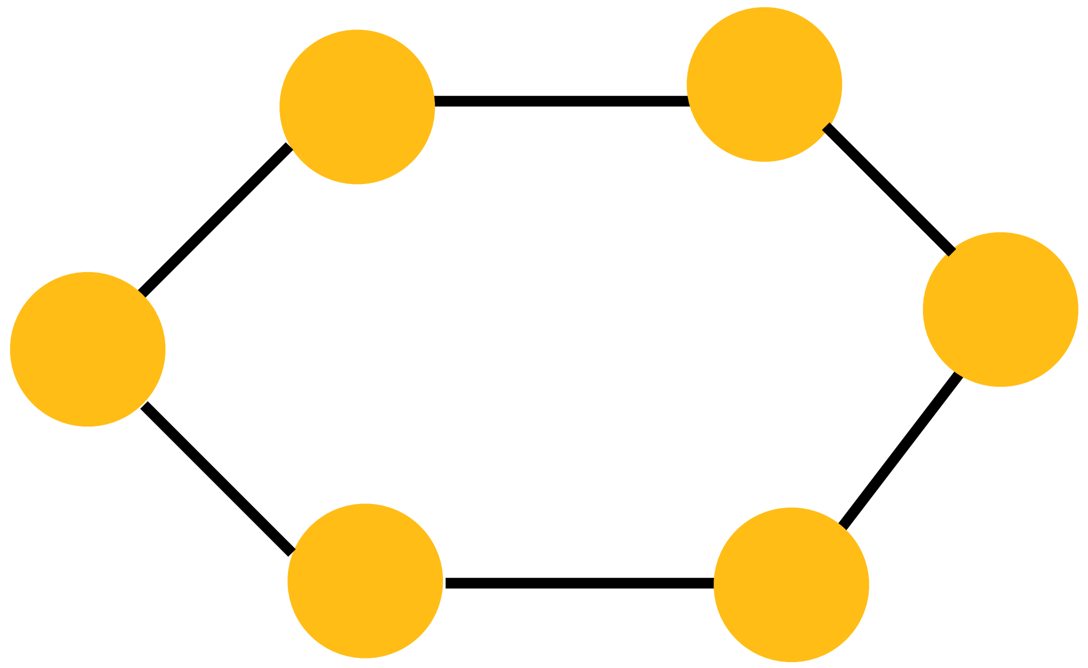
$$\mathbf{z}_G = \mathbf{z}_H \text{ if and only if } G \text{ is isomorphic to } H$$

Problem: This contains graph isomorphism testing, an NP-intermediate problem, where the best algorithm requires quasi-polynomial time (Babai, 2016).

Question: Where do MPNNs stand in graph distinguishability?

A Tale of Two Graphs



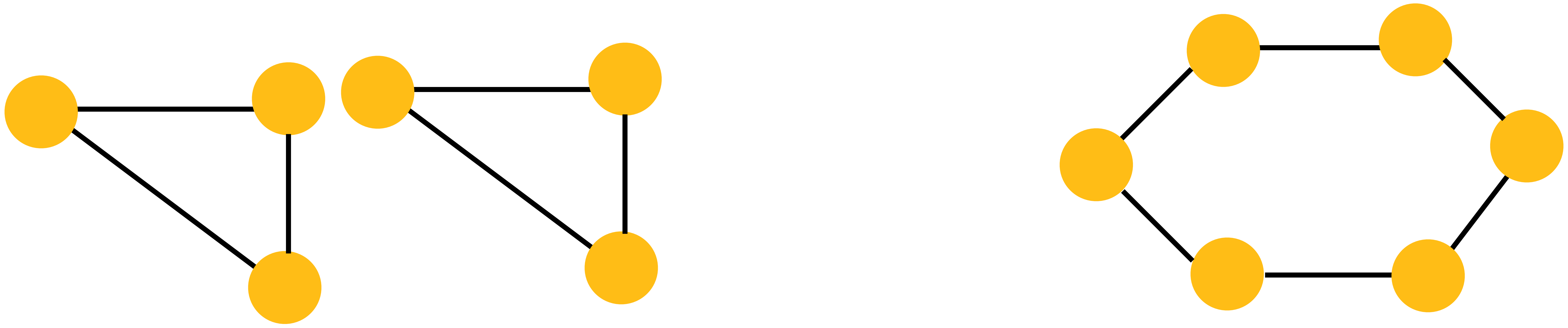


Problem: Any MPNN will learn identical representations for the graphs shown.

MPNNs cannot distinguish between two triangles and a 6-cycle — severe limitation for graph classification, as the predictions for these graphs will be identical regardless of the function we are trying to learn!

Is this only a problem for graph classification?

A Tale of Two Graphs

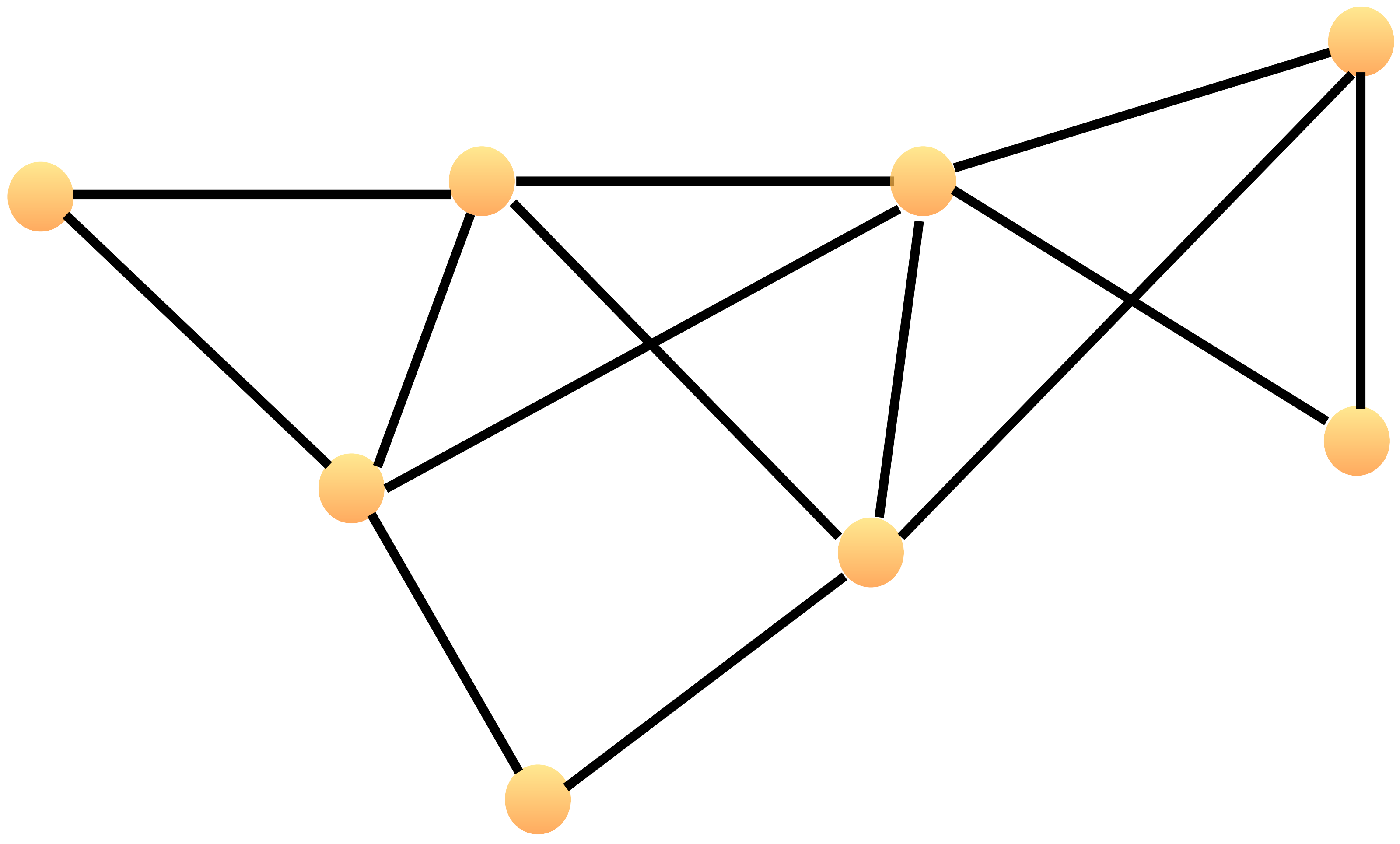


Task: A separator node has two neighbors that are non-adjacent. Consider the graph that is the disjoint union of graphs shown and classify the nodes as separator and non-separator.

All nodes in the 6-cycle are separator nodes, whereas all nodes in the triangles are non-separator nodes.

An MPNN will either predict all nodes to be separator nodes, or all of them as non-separator nodes, a random answer with exactly $50\,\%$ accuracy.

Finding the Culprits



Recall that we can embed a graph G using a multi-layer perceptron as follows:

$$f(G) = \mathrm{MLP}(\mathbf{A}^{G}_{[1]} \oplus \ldots \oplus \mathbf{A}^{G}_{[|V_G|]})$$

Problem: Order-dependent embedding of graphs - MLPs are expressive but lack relational inductive bias.

Message passing neural networks: Strong relational inductive bias, but not expressive.

Trade-off: Constrain the learning space (e.g., incorporating inductive bias), but not too much to entail strong limitations in the representation capacity.

Graph Isomorphism and Color Refinement

Graph Isomorphism

Two graphs G and H are isomorphic if there is a bijection between the vertex sets V_G and V_H:

$$f : V_G \mapsto V_H$$

such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ *and* $f(v)$ are adjacent in H.

We can restate this using features and matrices...

Two graphs G and H are isomorphic if and only if there exists a permutation matrix $\mathbf{P}$ such that:

$$\mathbf{P}\mathbf{A}^G\mathbf{P}^\top = \mathbf{A}^H \quad \text{and} \quad \mathbf{P}\mathbf{X}^G = \mathbf{X}^H.$$

where $\mathbf{A}^G$and $\mathbf{A}^H$ are the respective adjacency matrices and $\mathbf{X}^G$and $\mathbf{X}^H$ the respective node features.

Graph isomorphism testing: Problem of deciding whether the input graphs are isomorphic.

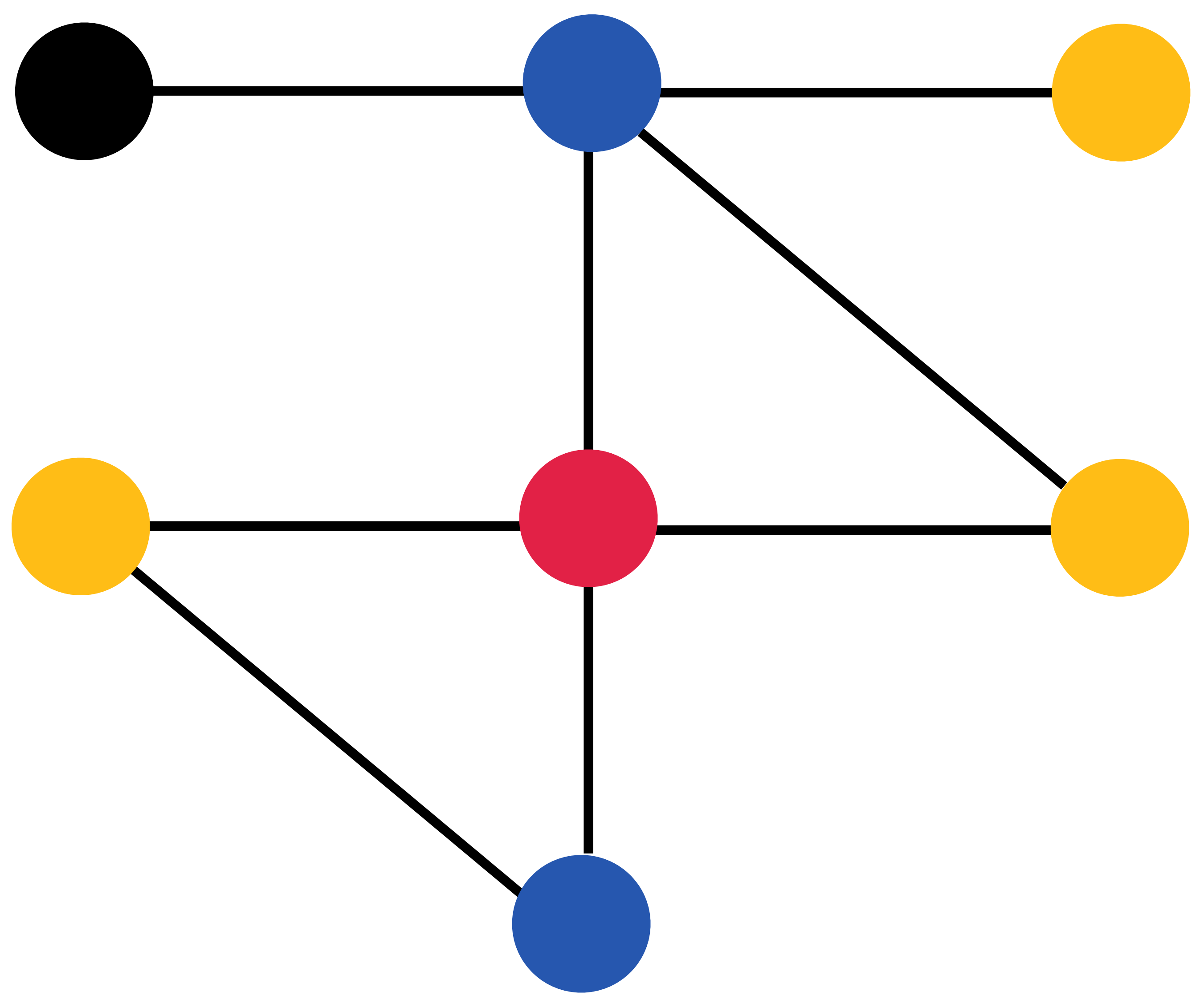
Exact testing: Suspected to be NP-intermediate - unsurprisingly beyond MPNNs.

Approximations: Many algorithms that can work well within broad classes of graphs.

Colour Refinement

Color refinement is a simple and effective algorithm for graph isomorphism testing:

1. **Initialization**: All vertices in a graph are initialized to their initial colors.
2. **Refinement**: All vertices are re-colored depending on their current color and the colors in their neighborhoods.
3. **Stop**: Terminate when the coloring stabilizes.



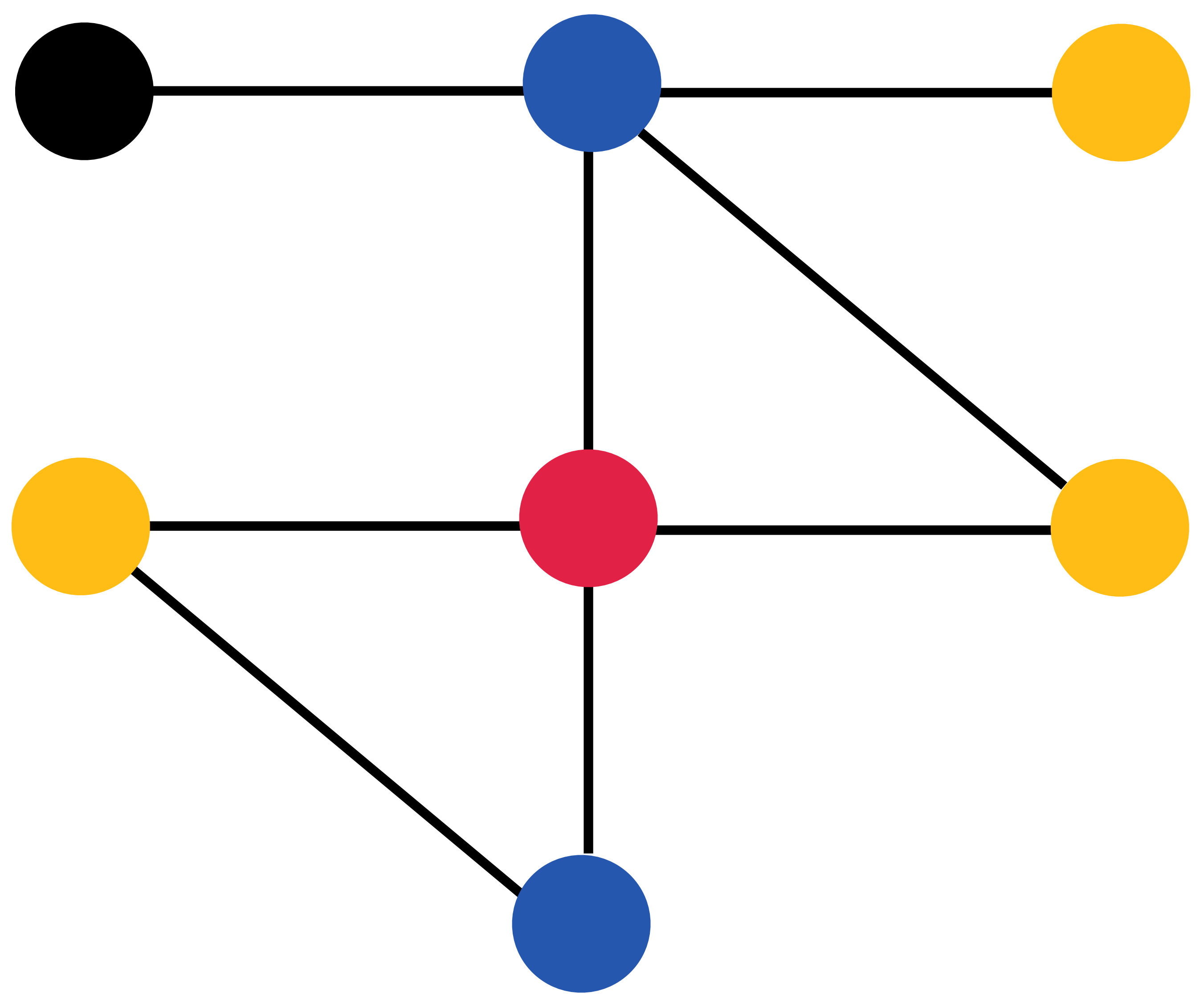
Colour Refinement

Given a graph $G = (V, E)$, and a set $\mathbf{C}$ of colors, a function

$$\lambda : V_G \mapsto \mathbf{C}$$

colors each vertex of the graph with a color from $\mathbf{C}$.

- **Partition**: Each λ induces a partition $\pi(\lambda)$ of V_G into vertex color classes.
- **Refinement** $\left(\pi(\lambda) \preceq \pi(\lambda')\right)$: A partition $\pi(\lambda)$ refines a partition $\pi(\lambda')$, if every element of $\pi(\lambda)$ is a subset of an element of $\pi(\lambda')$.
- **Stabilization** $\left(\pi(\lambda) \equiv \pi(\lambda')\right)$: If $\pi(\lambda) \preceq \pi(\lambda')$ and $\pi(\lambda') \preceq \pi(\lambda)$.



Colour Refinement

Input: A graph $G = (V, E)$ with an initial coloring $\lambda^{(0)}$.

1. **Initialization**: All vertices $u \in V$, are initialized to their initial colors $\lambda^{(0)}(u)$.

2. **Refinement**: All vertices $u \in V$ are recursively re-colored:

$$\lambda^{(i+1)}(u) = \mathrm{HASH}\Big(\lambda^{(i)}(u), \{\{\lambda^{(i)}(v) \mid v \in N(u))\}\}\Big),$$

   where double-braces denote a multiset, and HASH bijectively maps any pair (composed of a color and a multiset of colors) to a unique value in $\mathbf{C}$.

3. **Stop**: The algorithm terminates at iteration j, where j is the minimal integer satisfying:

$$\forall u, v \in V_G : \lambda^{(j+1)}(u) = \lambda^{(j+1)}(v) \text{ if and only if } \lambda^{(j)}(u) = \lambda^{(j)}(v).$$

Stopping condition is well-defined, since each iteration corresponds to a refinement, and there exists a minimal integer j such that $\pi(\lambda^{j}) \equiv \pi(\lambda^{(j+1)})$.

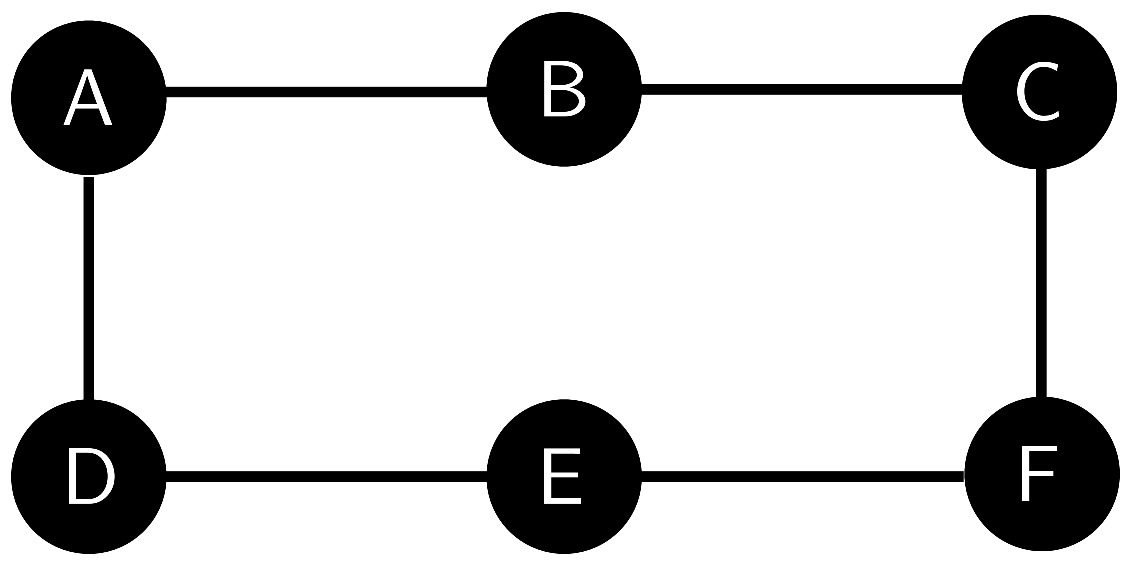
Colour Refinement

Colour refinement can be used to check whether two given graphs G and H are non-isomorphic:

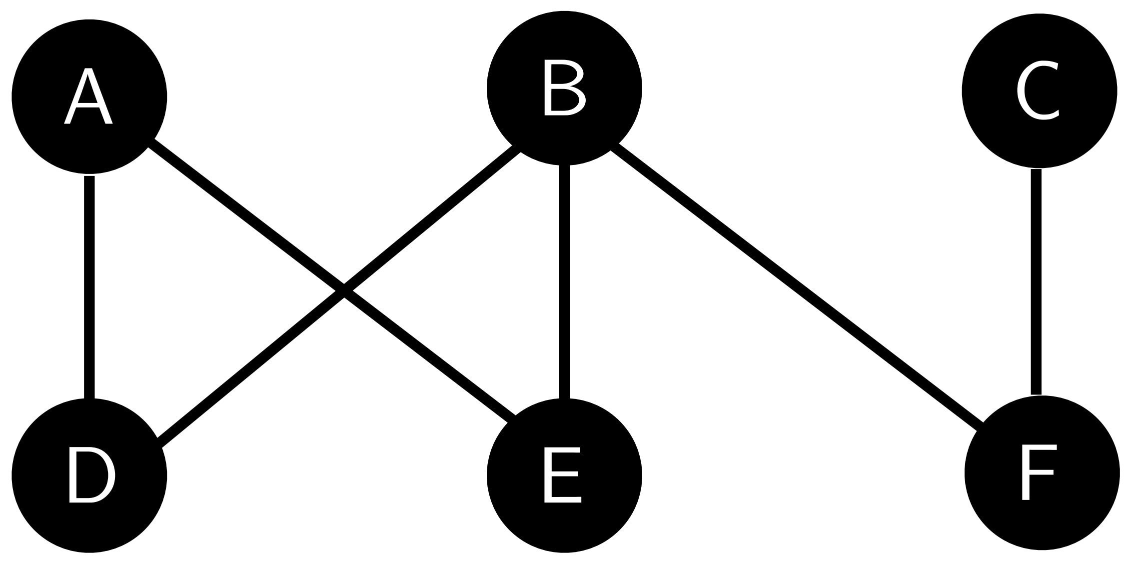
- Compute the stable coloring $\lambda^{(k)}$ on the disjoint union of G and H.
- If there is a $c \in \mathbf{C}$ in the stable coloring $\lambda^{(k)}$, where the numbers of vertices of color c differ in G and H, they are non-isomorphic.

Colour Refinement

G



H

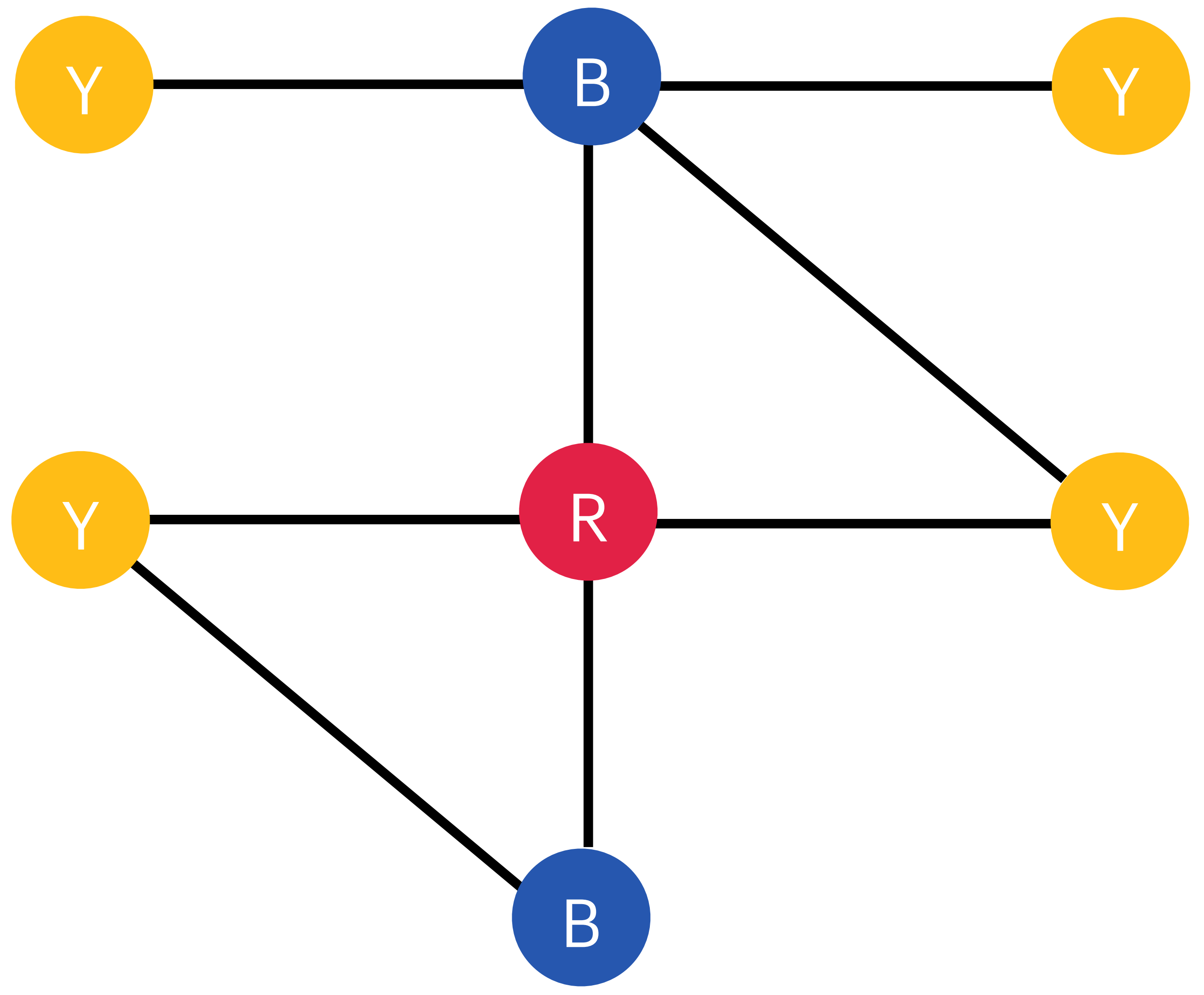


Soundness: Color refinement is sound for non-isomorphism checking: whenever it returns yes, for two graphs G and H, they are non-isomorphic.

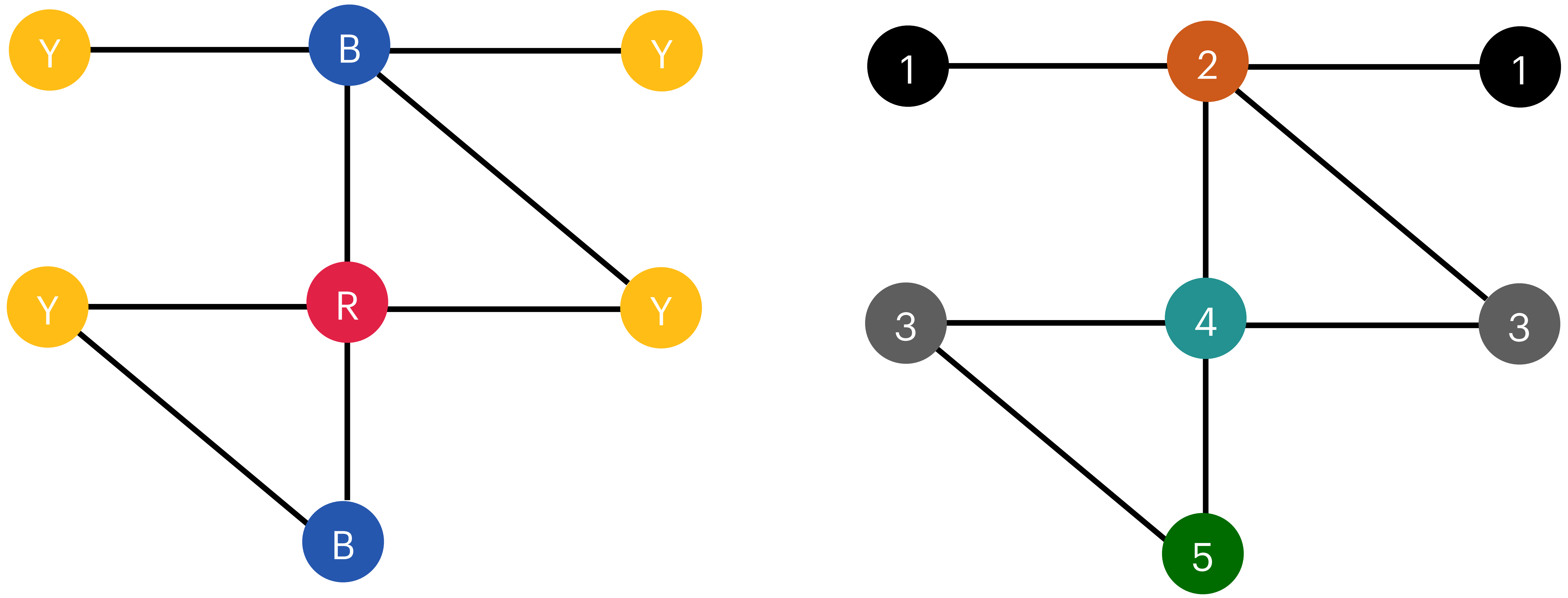
Incompleteness: Colour refinement is incomplete for non-isomorphism checking: even if G and H agree in every color class size in the stable coloring, the graphs might not be isomorphic.

Color refinement: AKA naive vertex refinement, or 1-dimensional Weisfeiler Lehman (1-WL) algorithm.

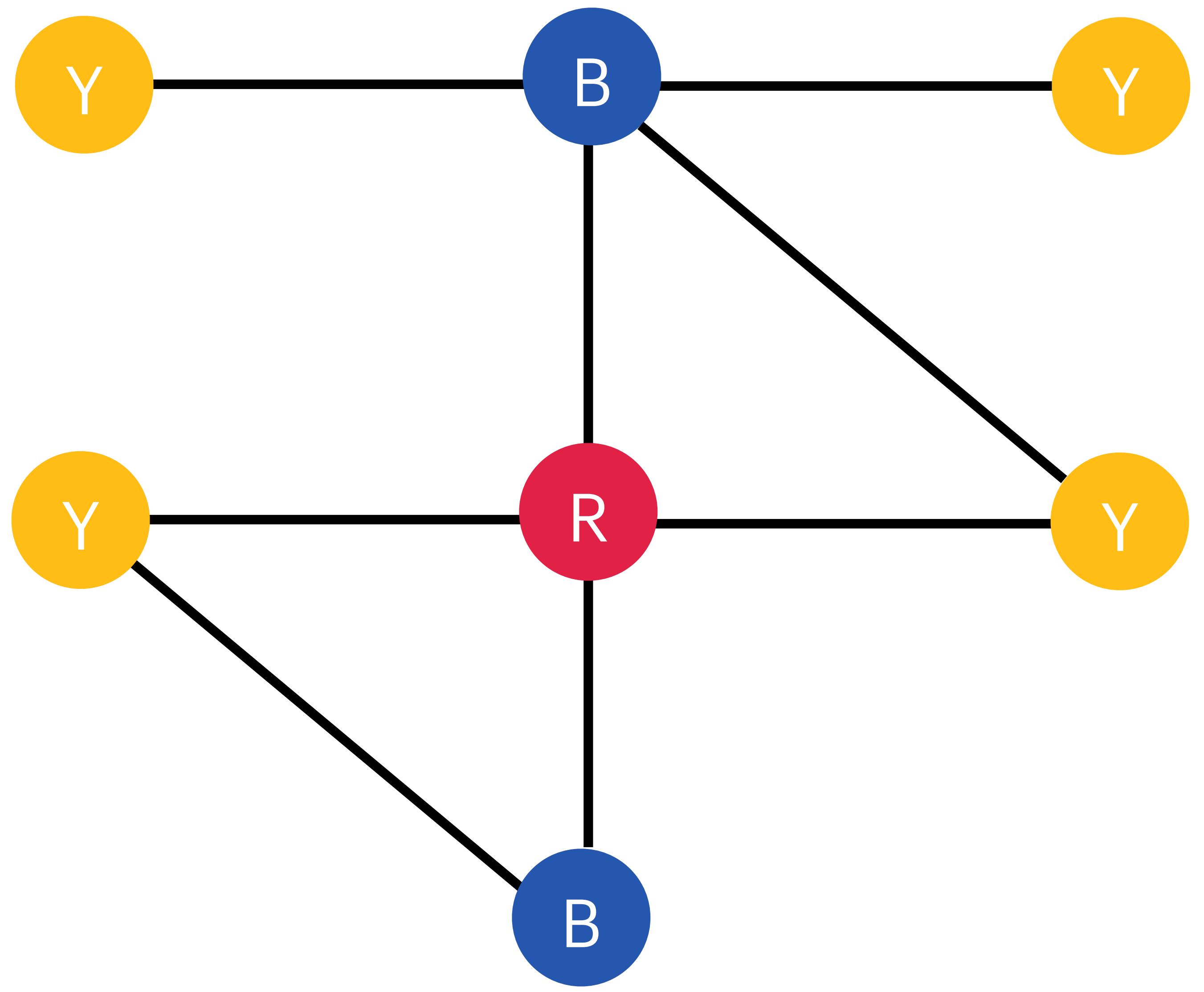
Colour Refinement: Example

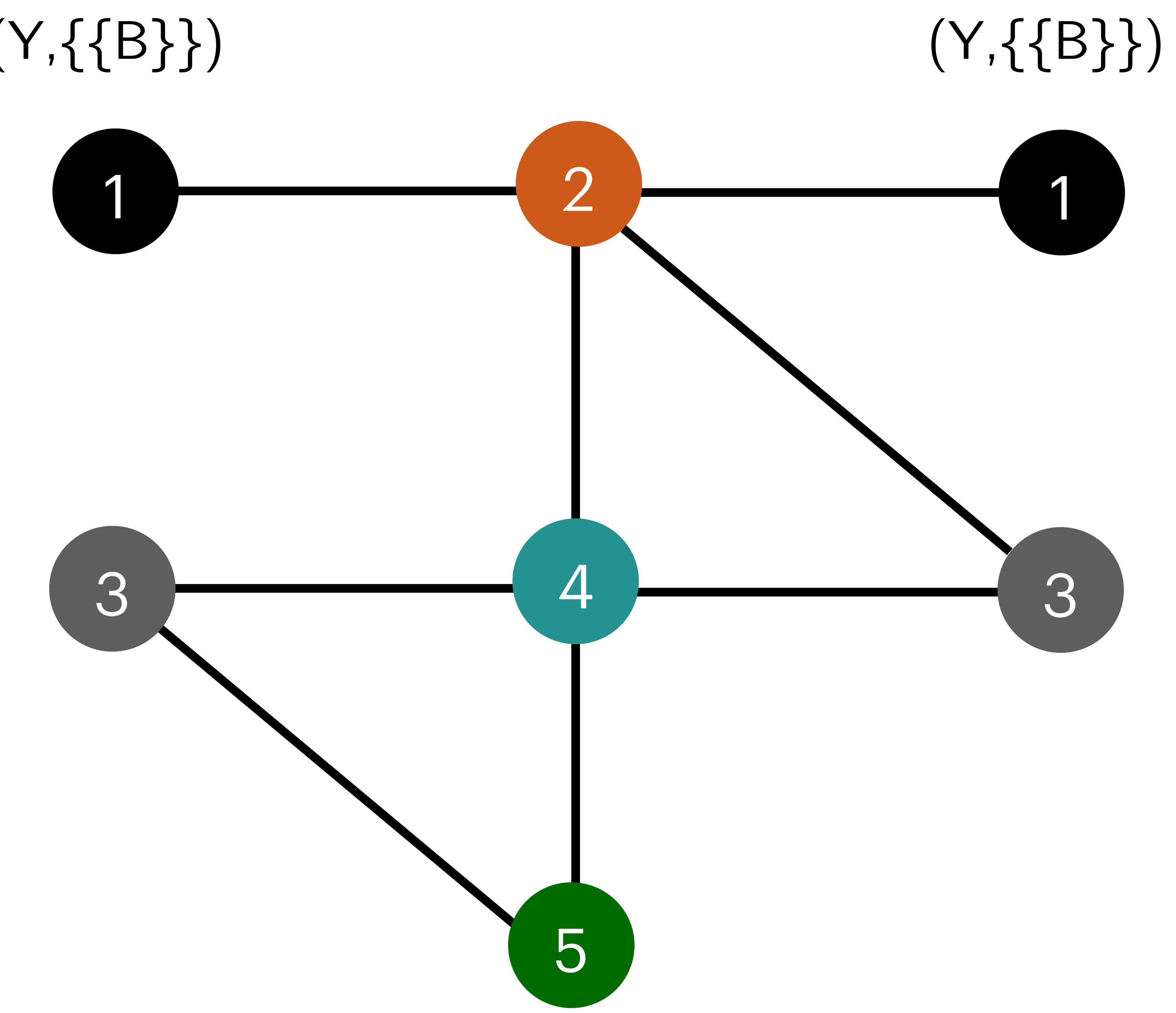


Colour Refinement: Example

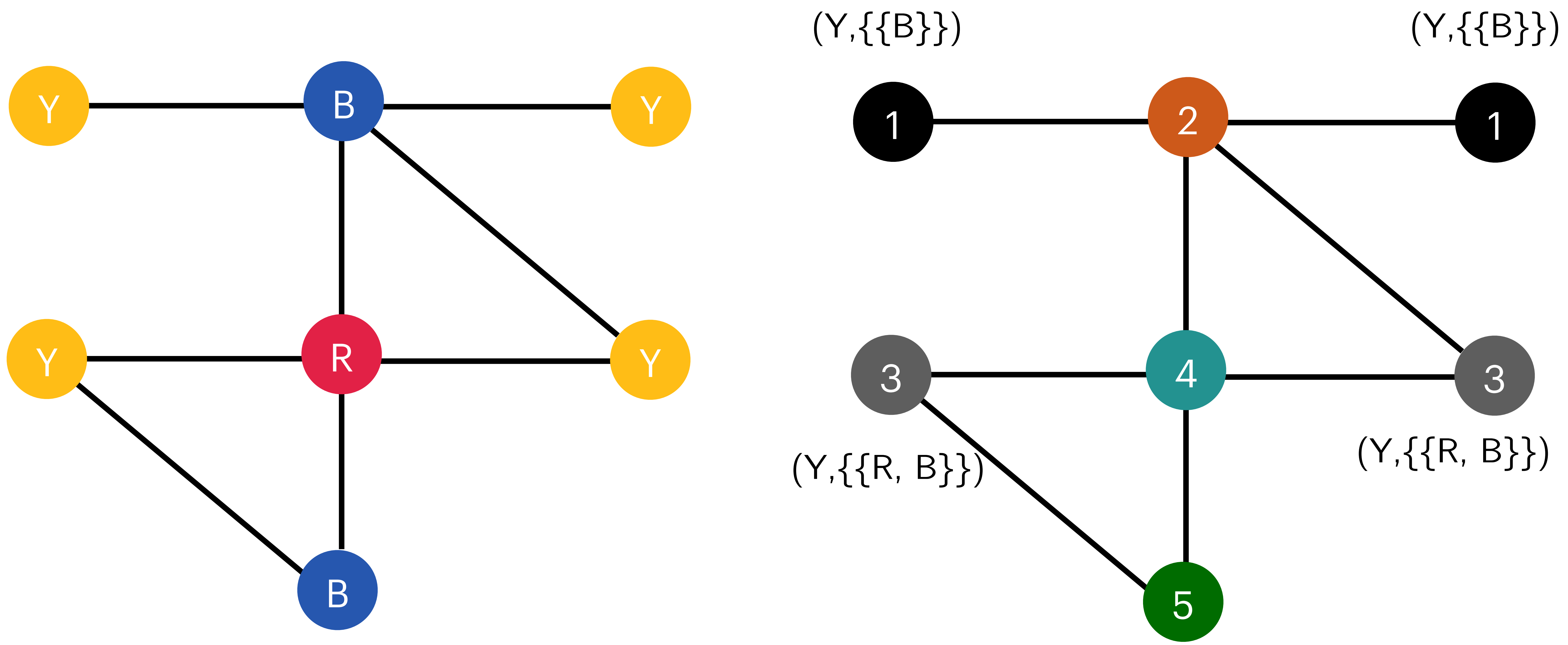


Colour Refinement: Example

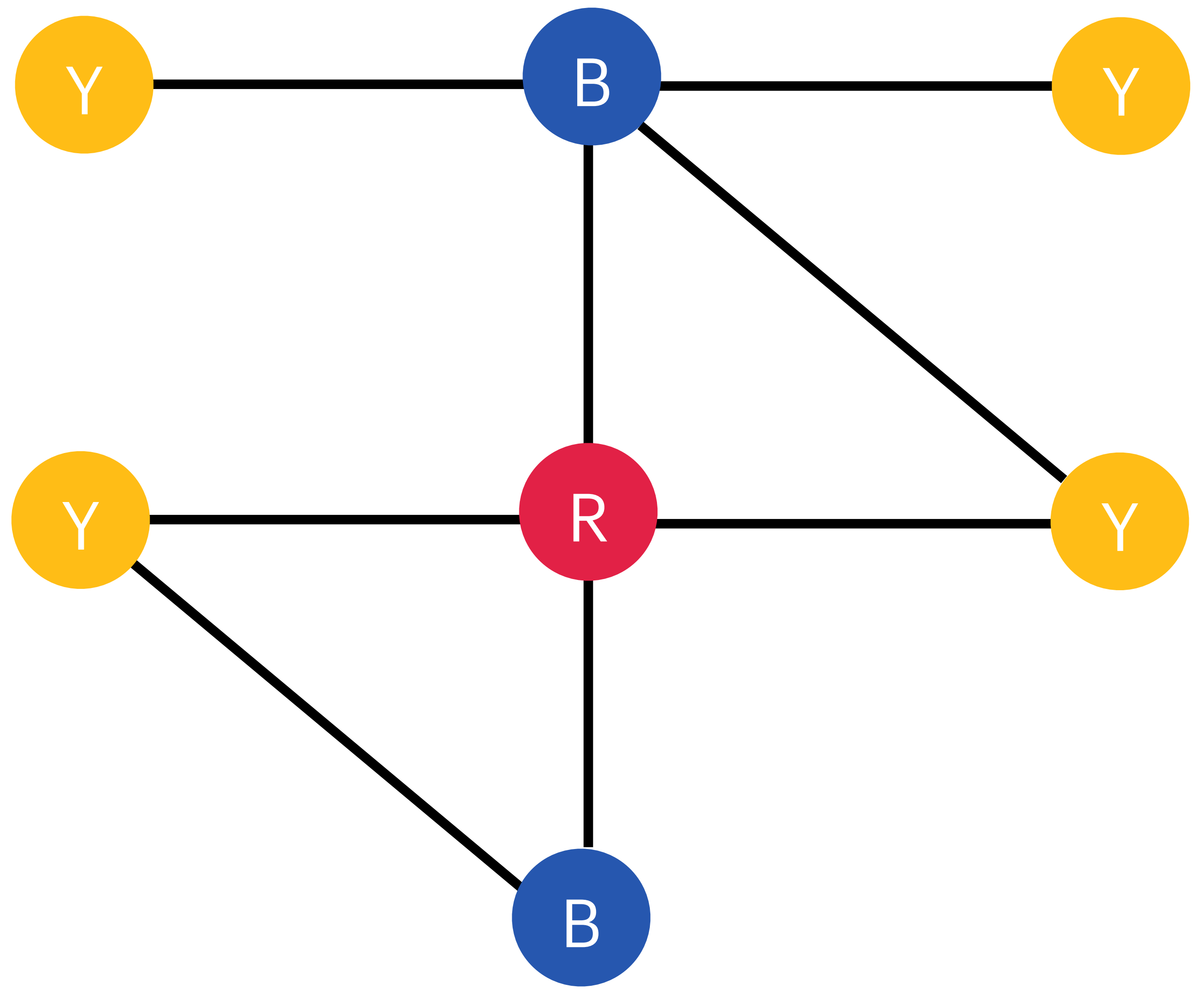


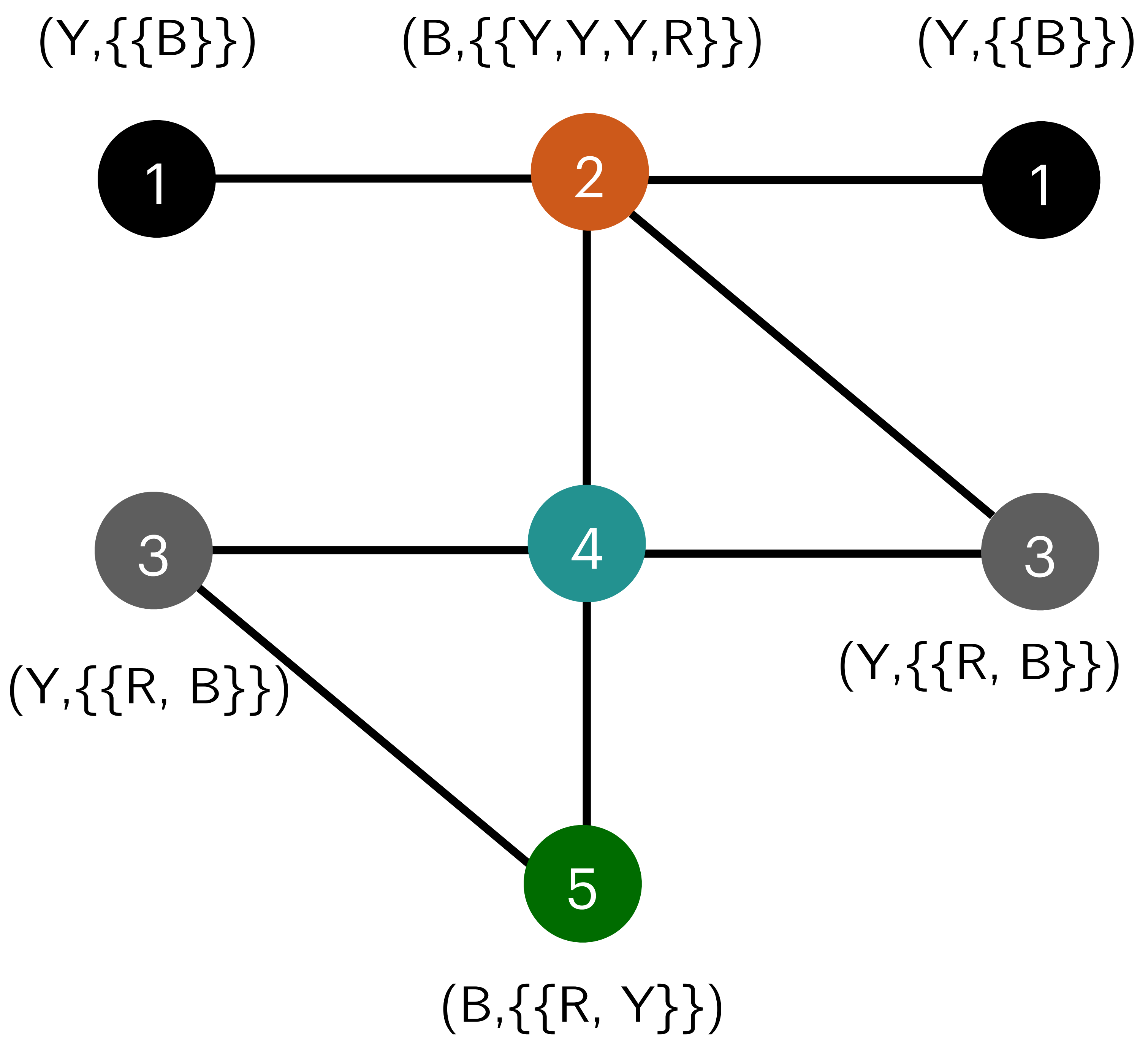


Colour Refinement: Example

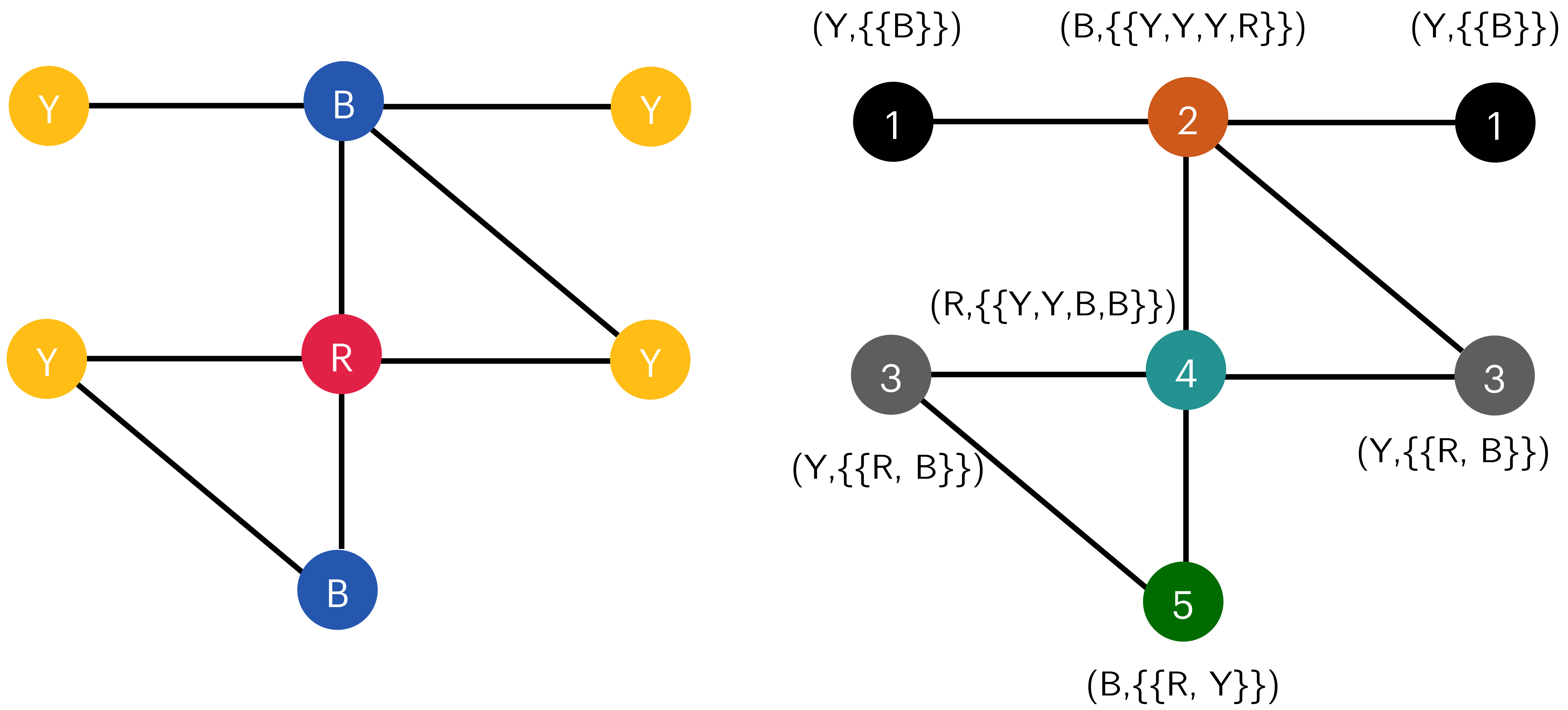


Colour Refinement: Example

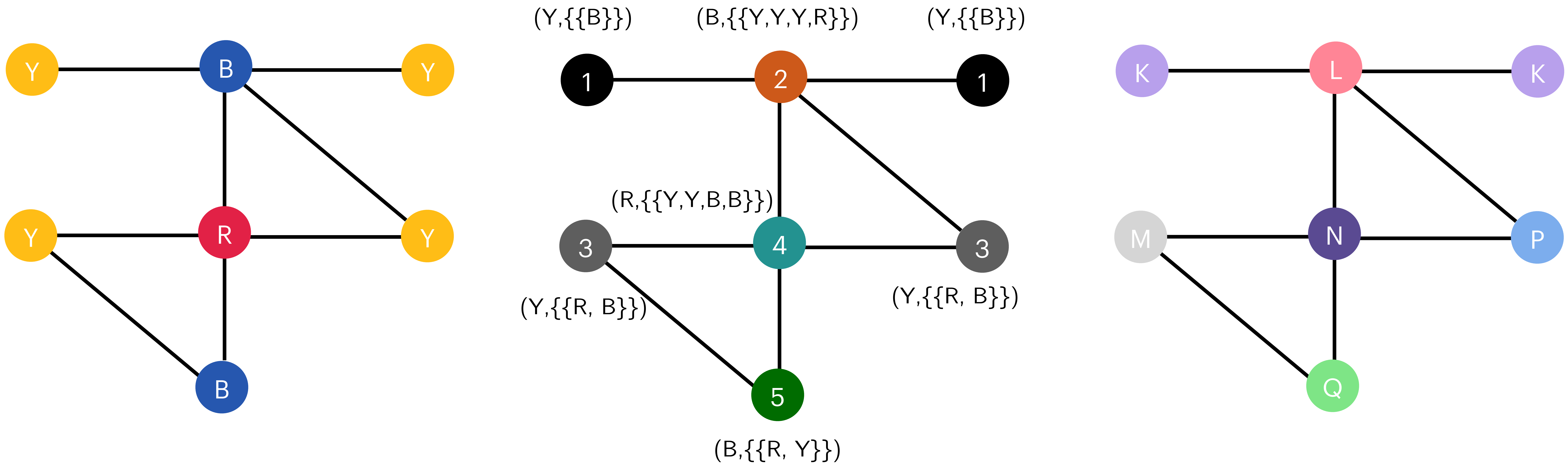




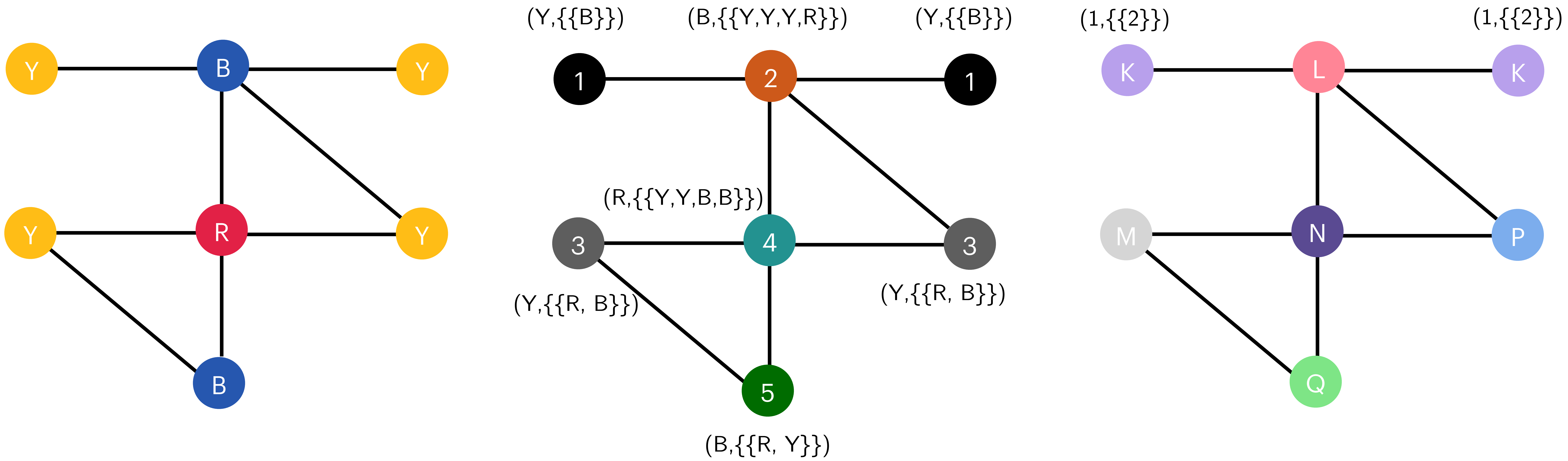
Colour Refinement: Example



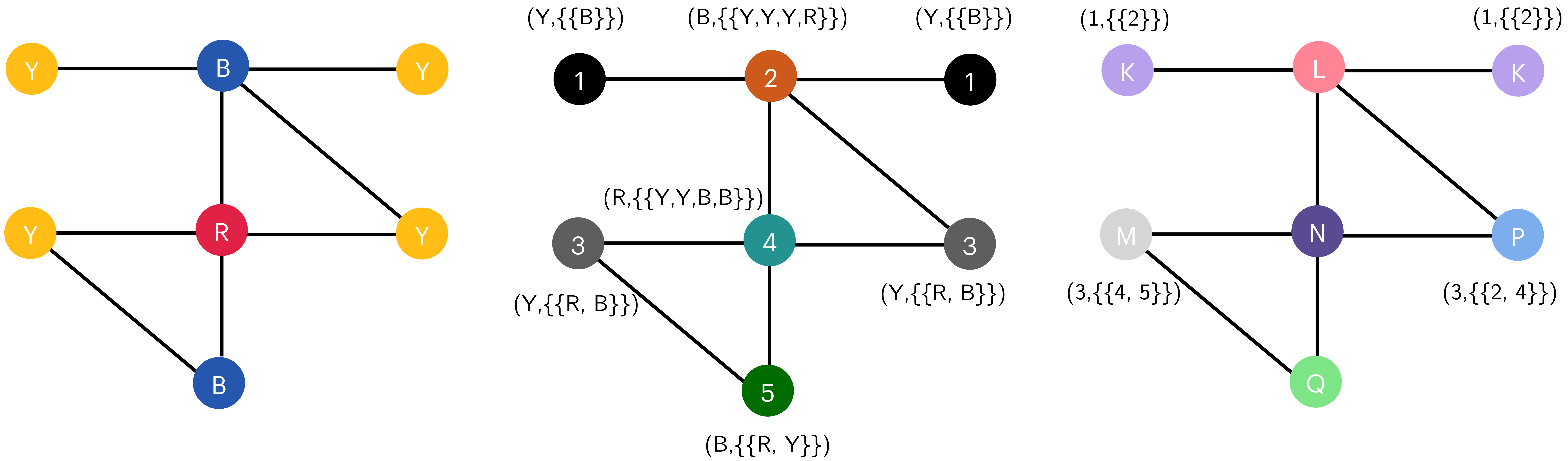
Colour Refinement: Example



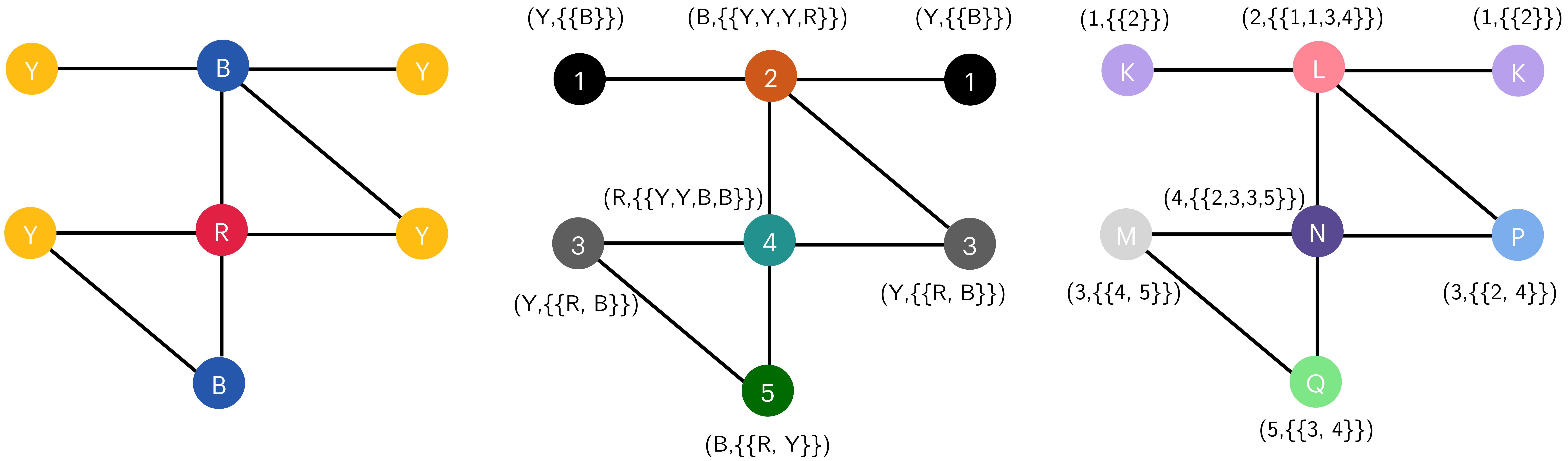
Colour Refinement: Example



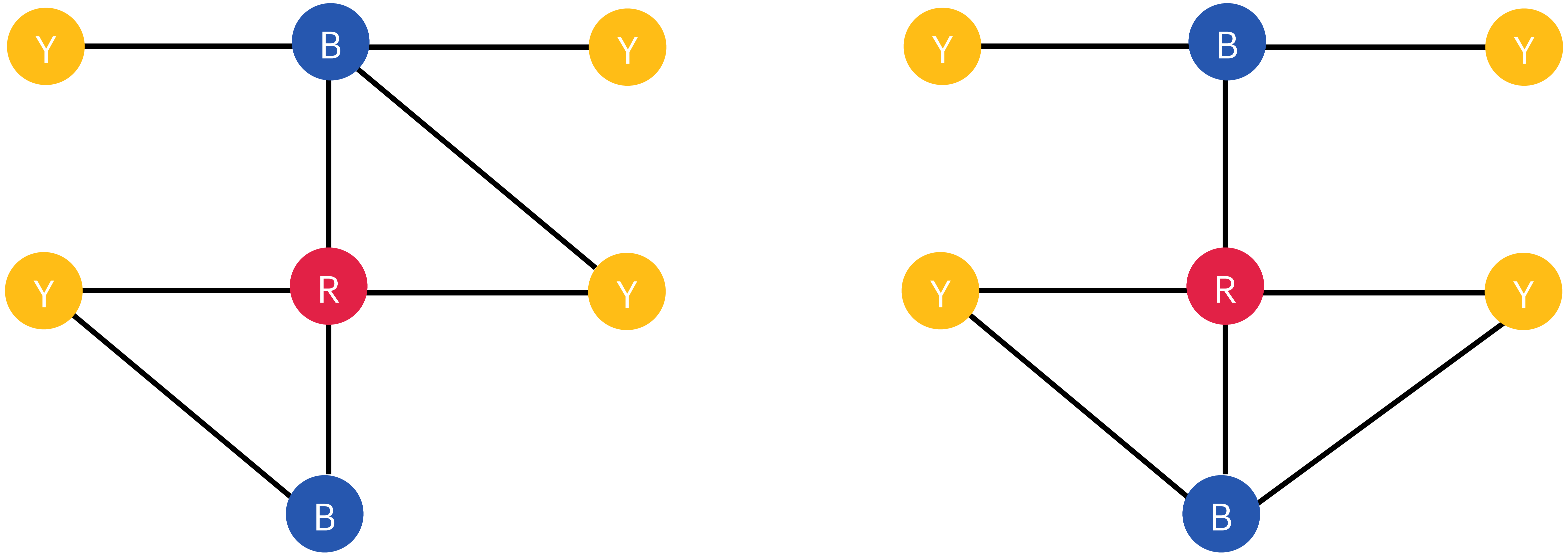
Colour Refinement: Example



Colour Refinement: Example



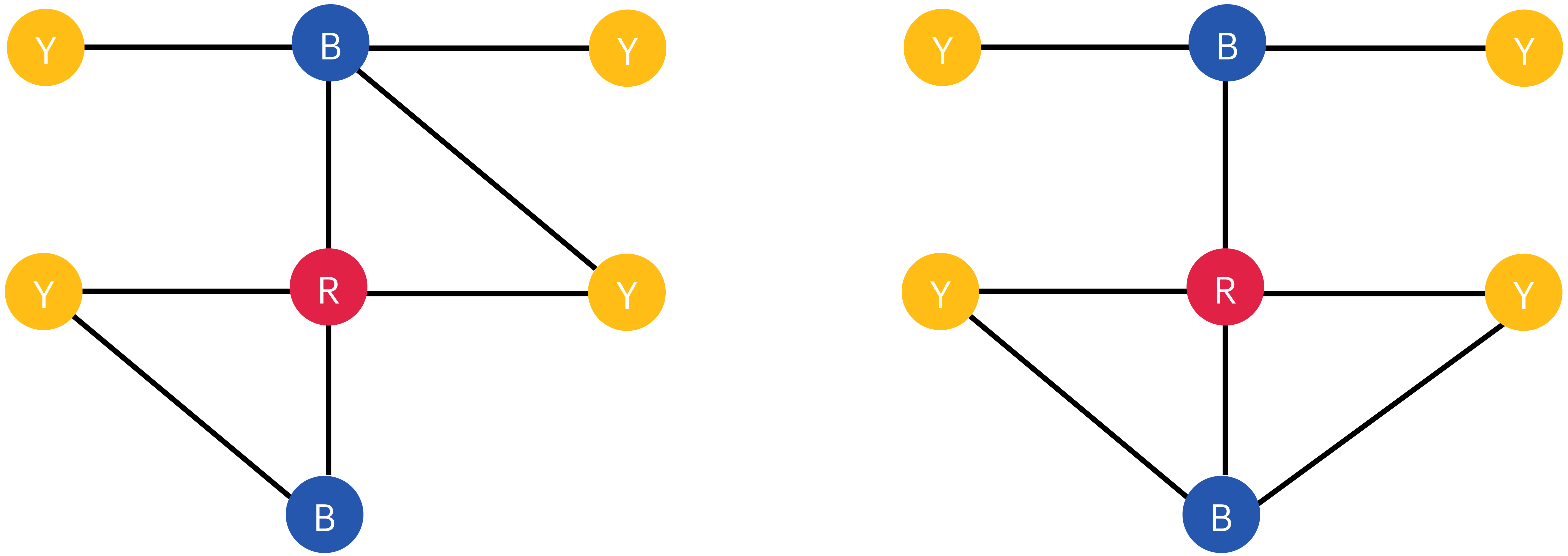
Colour Refinement: Example



Two graphs: Vertex color classes differ for these graphs - color refinement can distinguish...

Expressive Power of MPNNs

Colour Refinement: Example

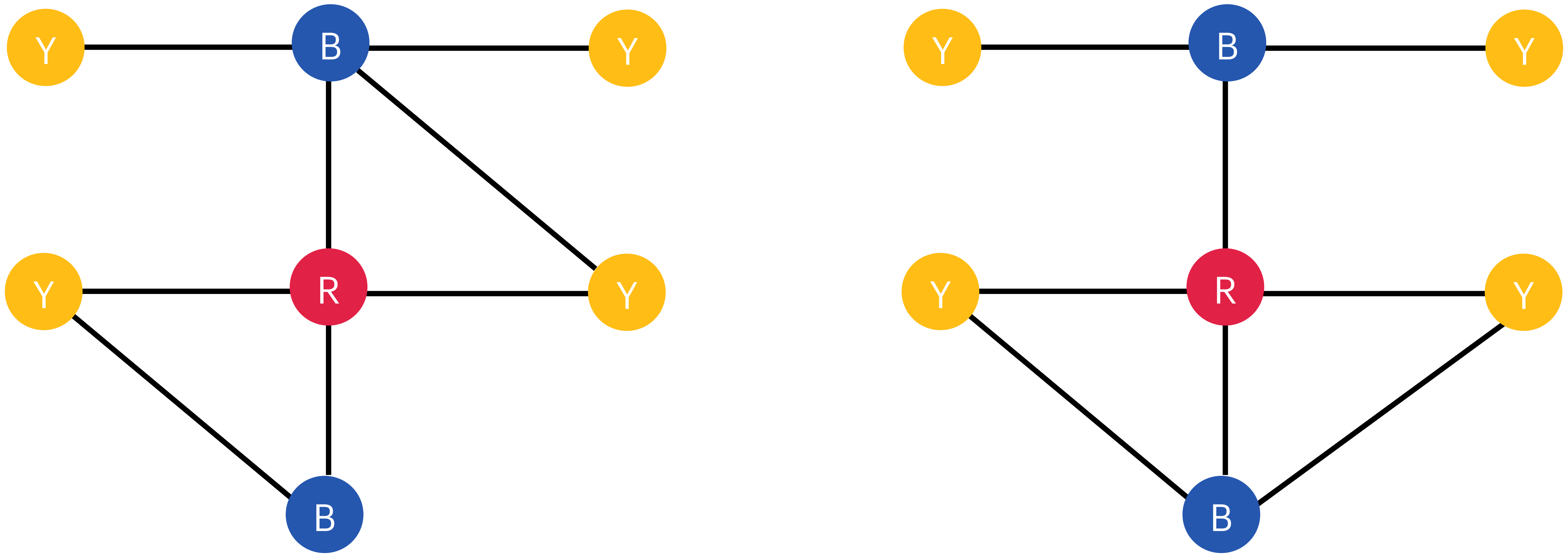


1-WL and neural message passing aggregate information from the neighborhoods and update accordingly:

$$\mathbf{h}_u^{(t)} = \mathit{combine}^{(t)}\left(\mathbf{h}_u^{(t-1)}, \mathit{aggregate}^{(t)}\left(\left\{\mathbf{h}_v^{(t-1)} \mid v \in N(u)\right\}\right)\right)$$

$$\lambda^{(i+1)}(u) = \mathsf{HASH}\left(\lambda^{(i)}(u), \{\{\lambda^{(i)}(v) \mid v \in N(u)\}\}\right)$$

Colour Refinement: Example

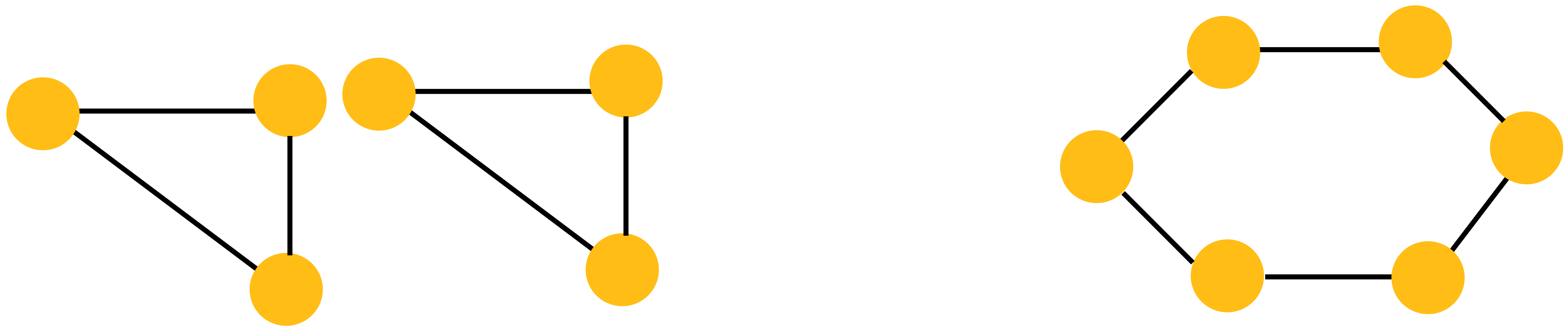


Can we view the rounds of the 1-WL algorithm as the layers of an MPNN?

$$\mathbf{h}_u^{(t)} = combine^{(t)}\Big(\mathbf{h}_u^{(t-1)}, aggregate^{(t)}\big(\{\mathbf{h}_v^{(t-1)} \mid v \in N(u)\}\big)\Big)$$

$$\lambda^{(i+1)}(u) = \mathsf{HASH}\big(\lambda^{(i)}(u), \{\{\lambda^{(i)}(v) \mid v \in N(u)\}\}\big)$$

An Upper Bound for Expressiveness of MPNNs

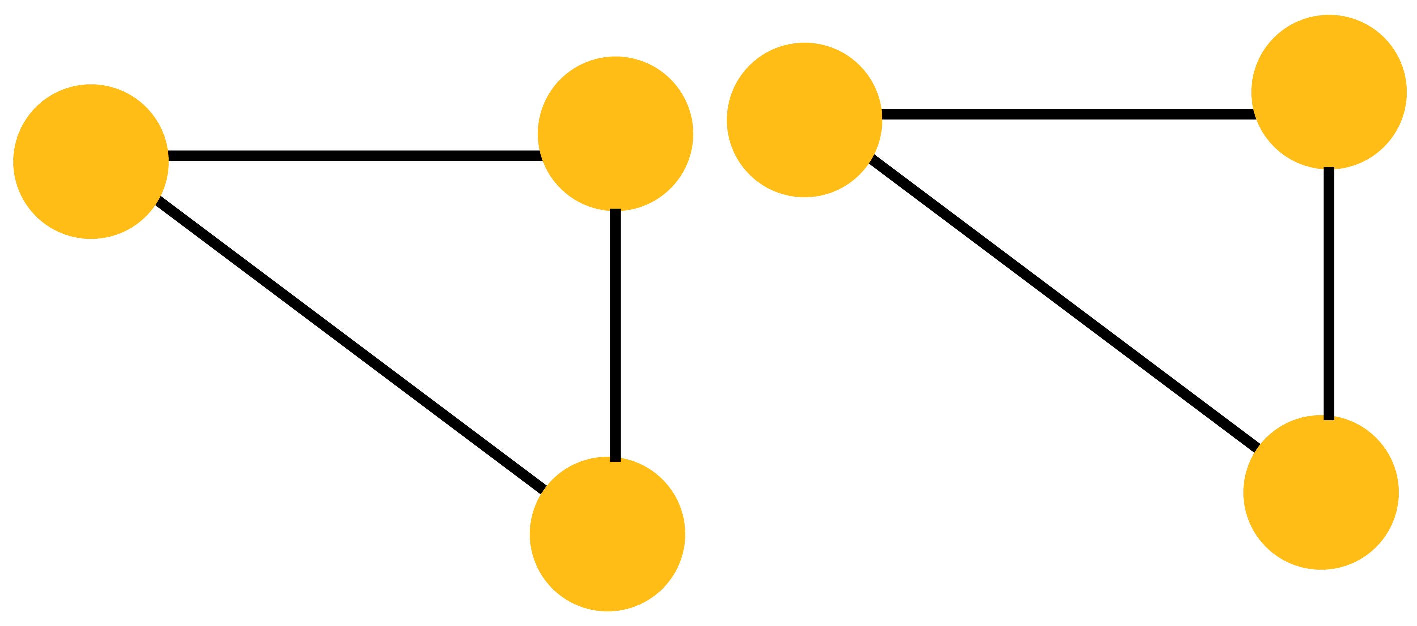


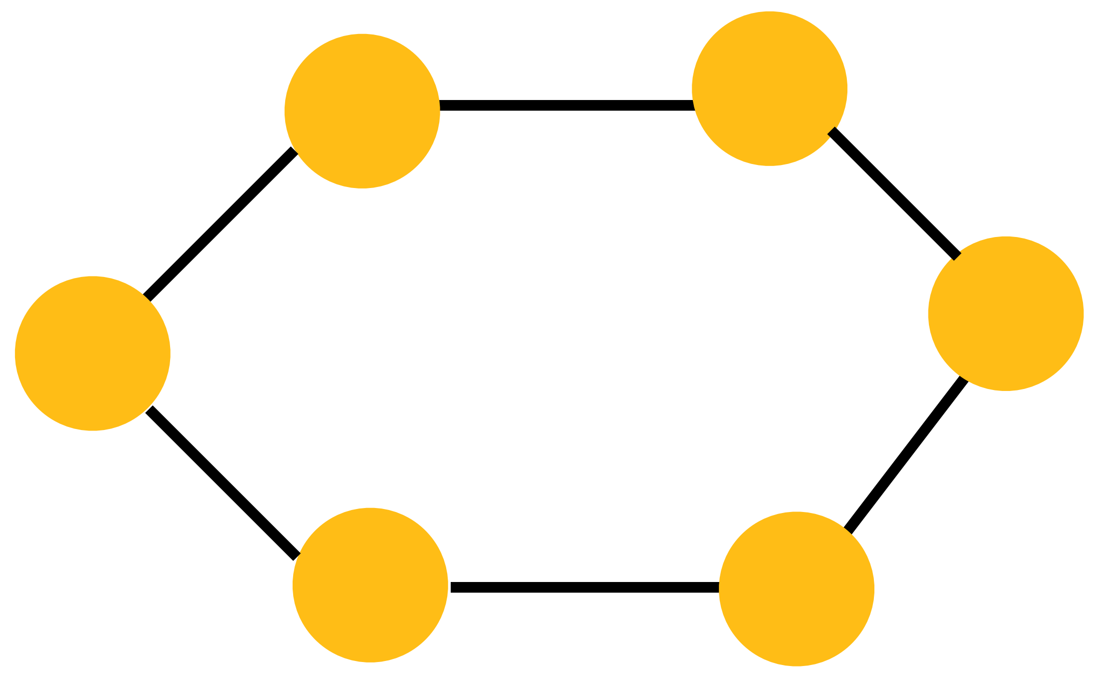
Theorem ([Morris et al., 2019, Xu et al., 2019]). Consider any MPNN that consists of k message-passing layers:

$$\mathbf{h}_u^{(t)} = \textit{combine}^{(t)}\Big(\mathbf{h}_u^{(t-1)}, \textit{aggregate}^{(t)}\big(\{\mathbf{h}_v^{(t-1)} \mid v \in N(u)\}\big)\Big)$$

Assuming only discrete input features $\mathbf{h}_u^{(0)} = \mathbf{x_u} \in \mathbb{Z}^d$, we have that $\mathbf{h}_u^{(k)} \neq \mathbf{h}_v^{(k)}$ only if the nodes u and v have different labels after k iterations of the 1-WL algorithm.

An Upper Bound for Expressiveness of MPNNs

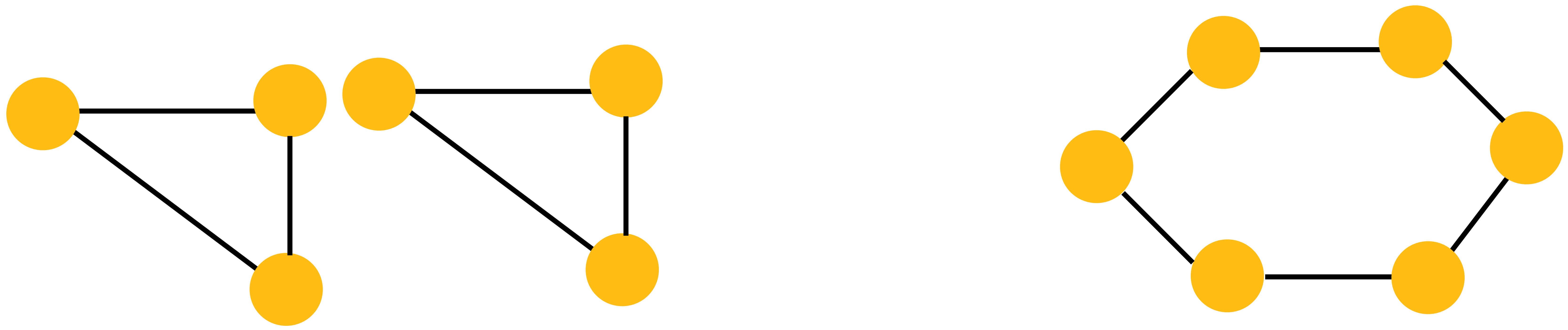




MPNNs are at most as powerful as the 1-WL test:

- If the 1-WL algorithm assigns the same label to two nodes, then any MPNN will also assign the same embedding to these two nodes.
- If the 1-WL test cannot distinguish between two graphs, then an MPNN is also incapable of distinguishing between these two graphs.

A Lower Bound for Expressiveness of MPNNs

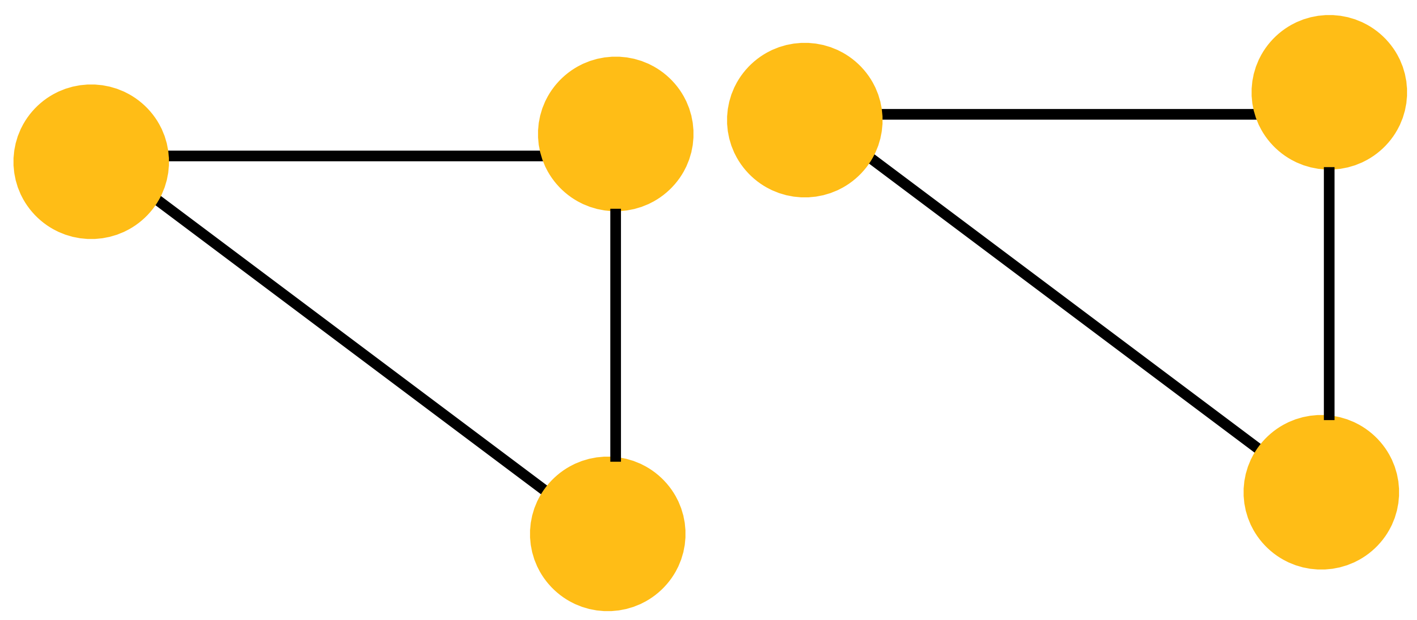


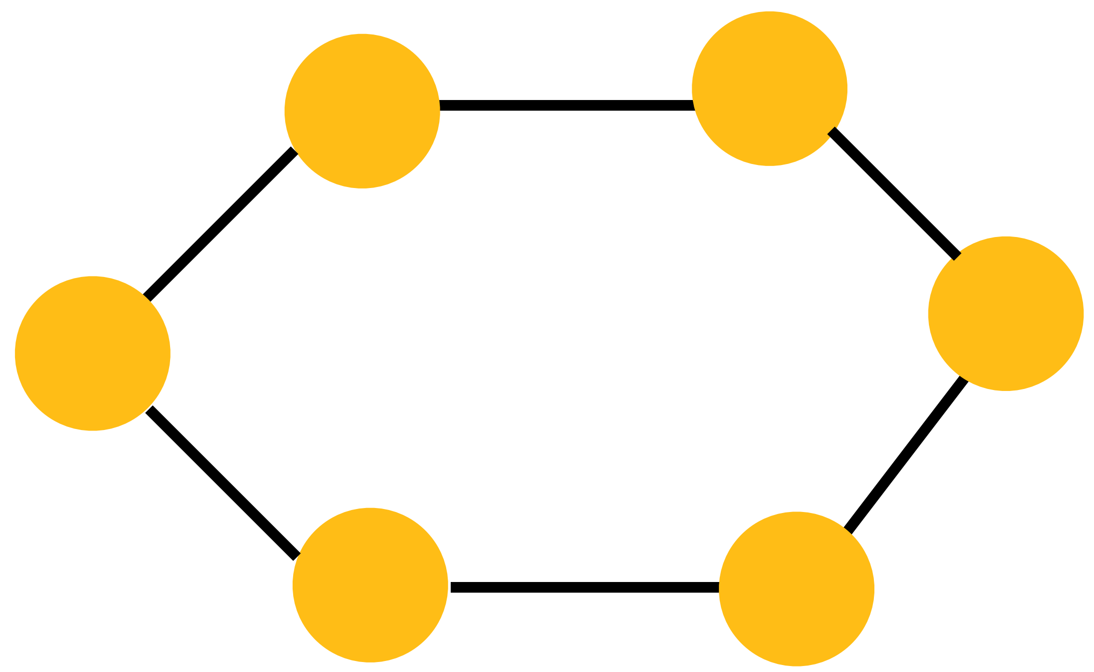
Theorem ([Morris et al., 2019, Xu et al., 2019]). There exists an MPNN such that $\mathbf{h}_u^{(k)} \neq \mathbf{h}_v^{(k)}$ if and only if the two nodes u and v have the same label after k iterations of the 1-WL algorithm.

In particular, the basic MPNN model is as powerful as 1-WL (in addition to GIN):

$$\mathbf{h}_u^{(t)} = \sigma\left(\mathbf{W}_{self}^{(t)}\mathbf{h}_u^{(t-1)} + \mathbf{W}_{neigh}^{(t)} \sum_{v \in N(u)} \mathbf{h}_v^{(t-1)}\right)$$

A Lower Bound for Expressiveness of MPNNs





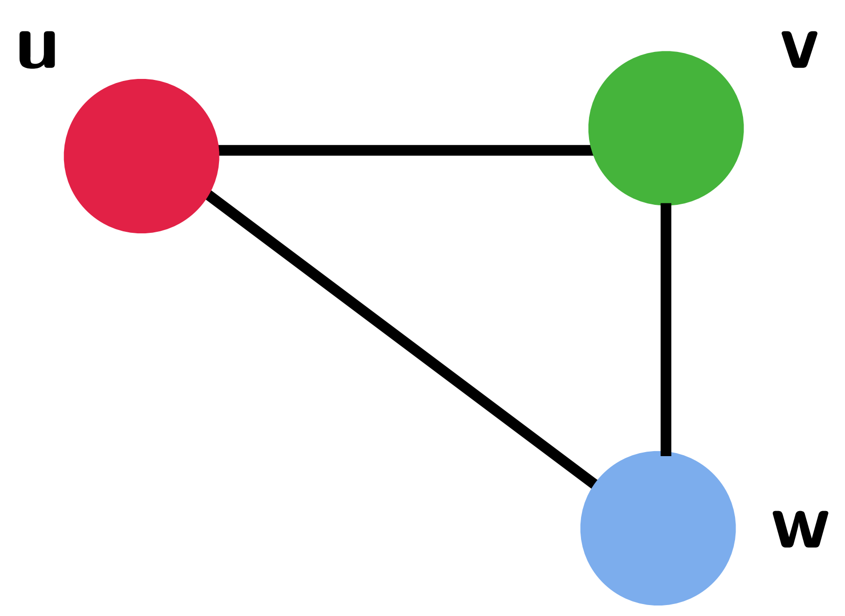
Most of the popular MPNN models, such as GCNs, are not even as expressive as 1-WL.

Key ingredient: The functions $aggregate^{(t)}$ and $combine^{(t)}$ need to be injective (Xu et al., 2019).

MPNNs are as powerful as 1-WL test under mild assumptions.

The Logic of Graphs

From Distinguishing Graphs to Capturing Functions



Question: Where do MPNNs stand in graph distinguishability?

Analysis: Expressive power through graph distinguishability: $\mathbf{z}_G = \mathbf{z}_H$ if and only if G is isomorphic to H

Result: MPNNs learnable, differentiable extension of the 1-WL with the same expressive power.

Question: What is the class of functions that is captured by MPNNs?

Idea: Characterizing classes of functions by a language...logic of graphs.

A Descriptive Complexity Perspective

WL hierarchy: The class of WL algorithms and forms an hierarchy, i.e., 1-WL, 2-WL,... as we shall see later.

Logic and WL: Connection between the WL hierarchy and first order logic with counting quantifiers:

Theorem (Cai et al., 1992). For all $k \geq 2$, two graphs G and H satisfy the same C^k-sentences if and only if $(k-1)$-WL does not distinguish them.

Together with the results of Morris et al. (2019) and Xu et al. (2019), this implies:

Proposition (Morris et al., 2019; Xu et al., 2019). Two graphs G and H are indistinguishable by all MPNNs if and only if they satisfy the same C^2 -sentences.

Remark: One may be tempted to think that this result entails that MPNNs can capture C^2: This result is about graph/node distinguishability, but we are interested characterizing the class of functions captured.

Territory of descriptive complexity — a branch of complexity theory, where the goal is to characterize complexity classes in terms of the logics that can capture the complexity classes (Immerman, 1995).

First-Order Logic: Syntax

Basics: A (first-order) relational vocabulary denoted by σ, consists of sets $\mathbf{R}$ of relation, $\mathbf{C}$ of constant, and $\mathbf{V}$ of variable names. A term is either a constant or a variable. An atom is of the form $P(s_1, \ldots, s_n)$, where P is an n-ary relation, and $s_1, \ldots, s_n$ are terms. A ground atom is an atom without variables.

Logical connectives and quantifiers: The logical connectives are negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$), and quantifiers are existential quantifier ($\exists$) and universal quantifier ($\forall$).

Formulas: First-order logic (FO) formulas are inductively built from atomic formulas using the logical constructors and quantifiers based on the grammar rule:

$$\Phi = \ P(s_1, \ldots, s_n) \ \mid \ \neg\Phi \ \mid \ \Phi \wedge \Phi \ \mid \ \Phi \vee \Phi \ \mid \ \exists x \,.\, \Phi \ \mid \ \forall x \,.\, \Phi,$$

where P is an n-ary relation, $s_1, \ldots, s_n$ are terms, and x is a variable.

Remark: Upper-case letters denote relation names, and lower case letters denote variables/constants.

First-Order Formulae

A variable x in a formula Φ is quantified, or bound if it is in the scope of a quantifier; otherwise, it is free.

A (first-order) sentence is a (first-order) formula without any free variables, also called a Boolean formula.

In the sequel, we write, e.g., Φ to denote Boolean formulas, and $\Phi(x_1, \ldots, x_k)$ to denote formulas with free variables $x_1, \ldots, x_k$.

As usual, some constructors are only syntactic sugar, i.e., we use usual abbreviations:

$$\forall x . \Phi \equiv \neg \exists x . \neg \Phi,$$

$$\Phi \vee \Psi \equiv \neg(\neg \Phi \wedge \neg \Psi),$$

$$\Phi \rightarrow \Psi \equiv \neg \Phi \vee \Psi,$$

and so we define the semantics based on the constructors $\neg$, $\wedge$, $\exists$.

First-Order Logic: Semantics

A first-order interpretation is a pair $I = (\Delta^I, \cdot^I)$, where Δ^I is a non-empty domain, and $\cdot^I$ is an interpretation function.

The interpretation function $\cdot^I$ maps every constant name a to an element $a^I \in \Delta^I$ of the domain, and every predicate name P with arity n to a subset $P^I \subseteq (\Delta^I)^n$ of the domain.

A variable assignment is a function $\mu : \mathbf{V} \mapsto \Delta^I$ that maps variables to domain elements.

Given an element $e \in \Delta^I$ and a variable $x \in \mathbf{V}$, we write $\mu[x \mapsto e]$ to denote the variable assignment that maps x to e, and that agrees with μ on all other variables.

For an interpretation I and a variable assignment μ, we define:

- $a^{I,\mu} = a^I$ for all constant names $a \in \mathbf{C}$,
- $x^{I,\mu} = \mu(x)$ for all variable names $x \in \mathbf{V}$,
- $P^{I,\mu} = P^I$ for all relation names $P \in \mathbf{R}$.

First-Order Logic: Semantics

Given an interpretation I and a variable assignment μ, the entailment relation ($\models$) is inductively defined as

- $I, \mu \models P(s_1, \ldots, s_n)$ if $(s_1^{I,\mu}, \ldots, s_n^{I,\mu}) \in P^{I,\mu}$,
- $I, \mu \models \neg\Phi(x_1, \ldots, x_n)$ if $I, \mu \not\models \Phi(x_1, \ldots, x_n)$,
- $I, \mu \models \Phi(x_1, \ldots, x_n) \land \Psi(y_1, \ldots, y_m)$ if $I, \mu \models \Phi(x_1, \ldots, x_n)$ and $I, \mu \models \Psi(y_1, \ldots, y_n)$,
- $I, \mu \models \exists x \,.\, \Phi(y_1, \ldots, y_n)$ if there exists $e \in \Delta^I$ such that $I, \mu[x \mapsto e] \models \Phi(y_1, \ldots, y_n)$,

Sentences: The truth value of sentences does not depend on any variable assignment; so, assignments are omitted in this case. We say that an interpretation I is a model of a sentence Φ if $I \models \Phi$.

Finite structures: An interpretation, or a model, is finite if its domain (or, universe) is finite. Our focus is on first-order logic over finite models/structures.

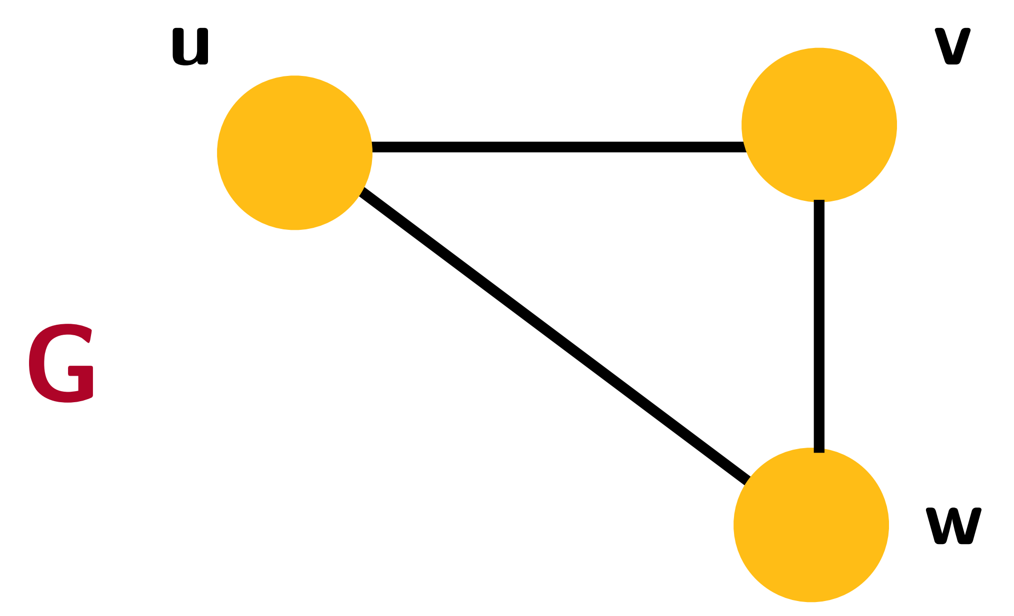
Unique names: We assume that constants are mapped to themselves (i.e., unique name assumption).

Logic of Graphs

The following FO formula with one free variable x:

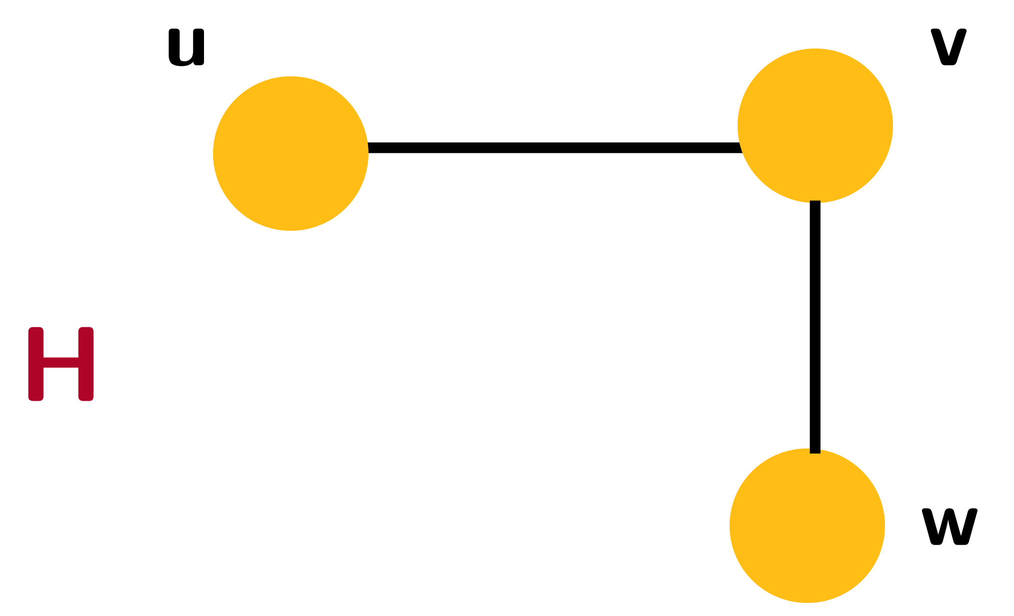
$$\Phi(x) = \exists y, z \; E(x, y) \wedge E(y, z) \wedge E(x, z) \wedge (x \neq z) \wedge (x \neq y) \wedge (y \neq z),$$

is in the language of graphs: $E(x, y)$ means that there is an edge between the nodes interpreting x and y.



Graphs as interpretations: View the graphs G and H as interpretations over a domain of nodes $\{u, v, w\}$:

- $E^G = \{(u, v), (v, w), (u, w)\}$
- $E^H = \{(u, v), (v, w)\}$



It is easy to verify that $G \vDash \Phi(u)$ and $H \nvDash \Phi(u)$.

The graph G is a model of $\Phi(x)$ when x is interpreted as u!

Logic of Colored Graphs

Colored graphs: The following FO formula

$$\Psi(x) = Red(x) \wedge \exists y\big(E(x,y) \wedge Blue(y) \wedge \exists z\big(E(x,z) \wedge Green(z)\big)\big)$$

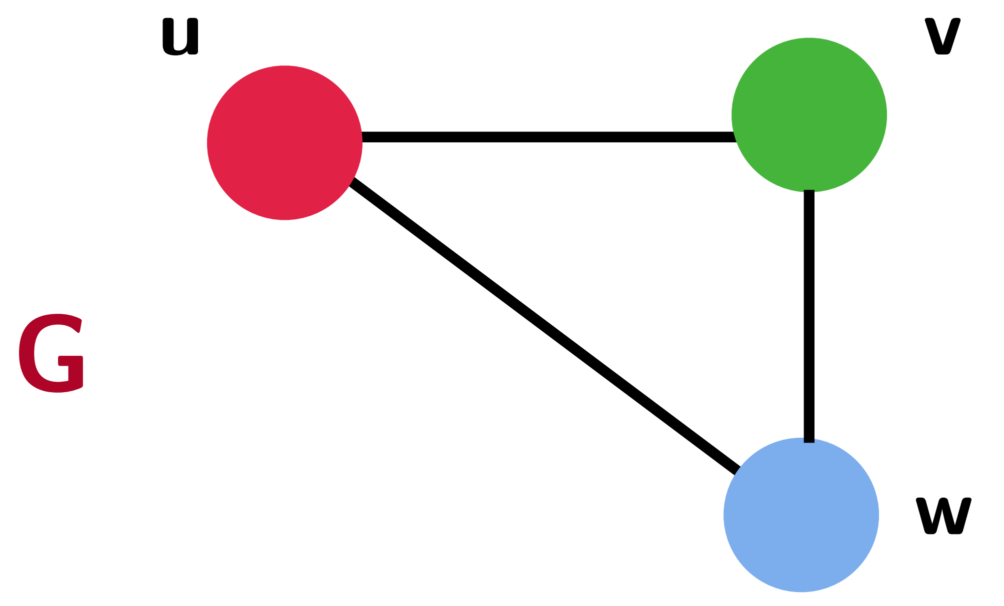
requires a red node connected to a blue and a green node in the input graph to satisfy the specified property:

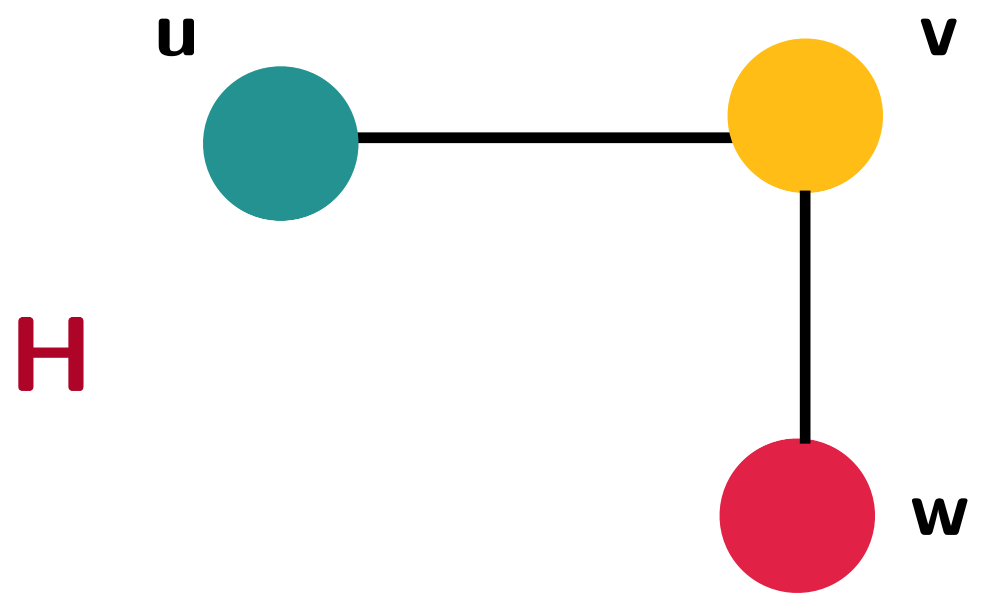
$$G \models \Psi(u) \text{ and } H \not\models \Psi(u)$$

We are interested in C^2, i.e., FO^2 extended with counting quantifiers:

$$\Theta(x) = \neg \exists^{\geq 3} y\big(Red(y) \wedge E(x,y) \wedge \exists^{\geq 5} x E(y,x)\big).$$

A graph G satisfies $\Theta(v)$ if and only if v has at most 2 red neighbors in G that have degree at least 5.





Two-Variable Fragment of First-Order Logic

FO^k: k-variable fragment of first-order logic. The formula from earlier is in FO^3:

$$\Psi(x) = Red(x) \land \exists y \big(E(x,y) \land Blue(y) \land \exists z \big(E(x,z) \land Green(z) \big) \big)$$

This reduces their expressive power: FO^2 is strictly contained in FO, i.e, there are FO formulas not in FO^2.

Re-using variables: $\Psi(x)$ can be equivalently written (by re-using the variable y in place of z) in FO^2:

$$\Psi(x) = Red(x) \land \exists y \big(E(x,y) \land Blue(y) \land \exists y \big(E(x,y) \land Green(y) \big) \big)$$

Remark: C is a syntactic extension of FO, as counting quantifiers of the form $\exists^{\geq k} x$ can be simulated with standard existential quantifiers using k variables. Counting quantifiers add expressiveness to FO^2.

Logical Characterization of MPNNs

A Logical Characterization for MPNNs

Question: What is the class of functions that is captured by MPNNs (Barcelo et al 2020)?

Context: Node classification and Boolean functions.

A logical node classifier is a formula $\Phi(x)$ in C^2 with exactly one free variable

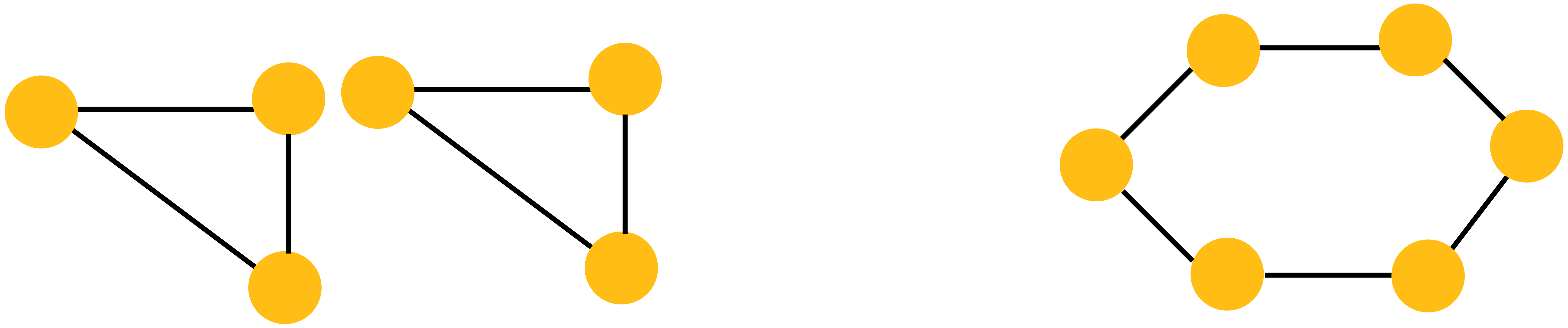
$$\Phi(u) : V_G \mapsto \mathbb{B} \text{ for each node } u \in V_G$$

An MPNN classifier M captures a logical classifier $\Phi(x)$ when both classifiers coincide over every input: if for every graph G and node u in G, it holds that $M(G, v)$ evaluates to true if and only if $G \models \Phi(u)$.

An MPNN classifier M captures a logic $\mathscr{L}$ if for every $\Phi(x) \in \mathscr{L}$, there exists an MPNN that captures $\Phi(x)$.

Goal: Identify a logic that is captured by MPNNs — identifying the expressive power of MPNNs.

A Logical Characterization for MPNNs



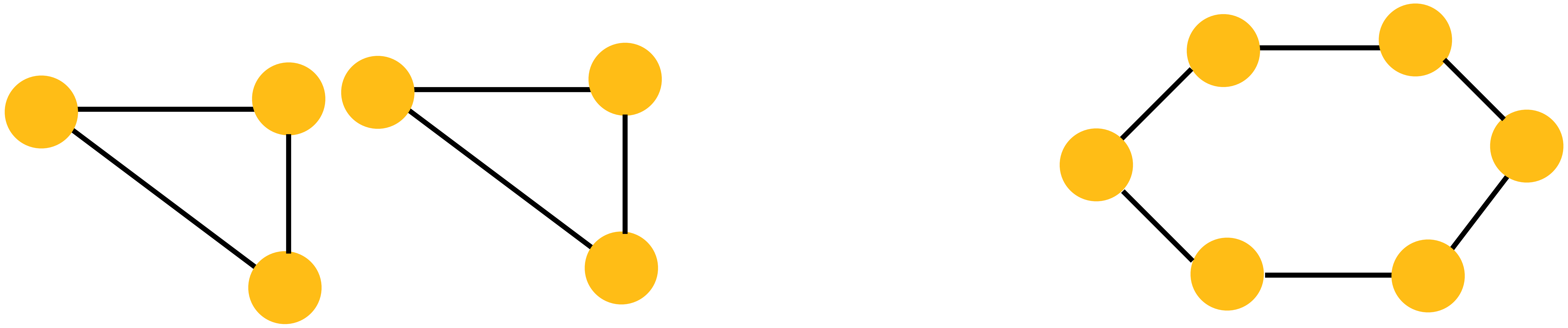
Theorem (Barcelo et al., 2020). Each C^2 classifier can be captured by an MPNN with global readout:

$$\mathbf{h}_u^{(t)} = combine^{(t)}\Big(\mathbf{h}_u^{(t-1)}, aggregate^{(t)}\big(\{\!\!\{\mathbf{h}_v^{(t-1)} \mid v \in N(u)\}\!\!\}\big), read^{(t)}\big(\{\!\!\{\mathbf{h}_w^{(t-1)} \mid w \in G\}\!\!\}\big)\Big).$$

The following formula cannot be expressed in C^2:

$$\Phi(x) = \exists y, z\ E(x,y) \wedge E(y,z) \wedge E(x,z) \wedge (x \neq z) \wedge (x \neq y) \wedge (y \neq z)$$

A Logical Characterization for MPNNs



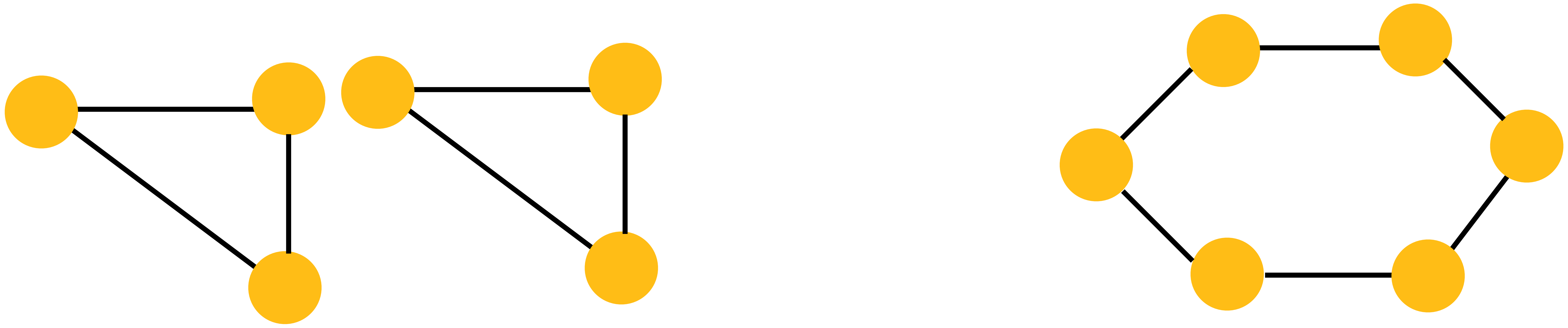
Theorem (Barcelo et al., 2020). Each C^2 classifier can be captured by an MPNN with global readout:

$$\mathbf{h}_u^{(t)} = \mathit{combine}^{(t)}\Big(\mathbf{h}_u^{(t-1)}, \mathit{aggregate}^{(t)}\big(\{\mathbf{h}_v^{(t-1)} \mid v \in N(u)\}\big), \mathit{read}^{(t)}\big(\{\mathbf{h}_w^{(t-1)} \mid w \in G\}\big)\Big).$$

Size of the network: The depth of the MPNN is bounded by the depth of the formula.

Special cases: Result holds even for homogeneous MPNNs and also for MPNNs with a single (final) global readout, but in the latter case we require MPNN to be non-homogeneous.

A Logical Characterization for MPNNs



MPNNs without any readouts can capture graded modal logic, a strict subset of C^2 (Barcelo et al., 2020).

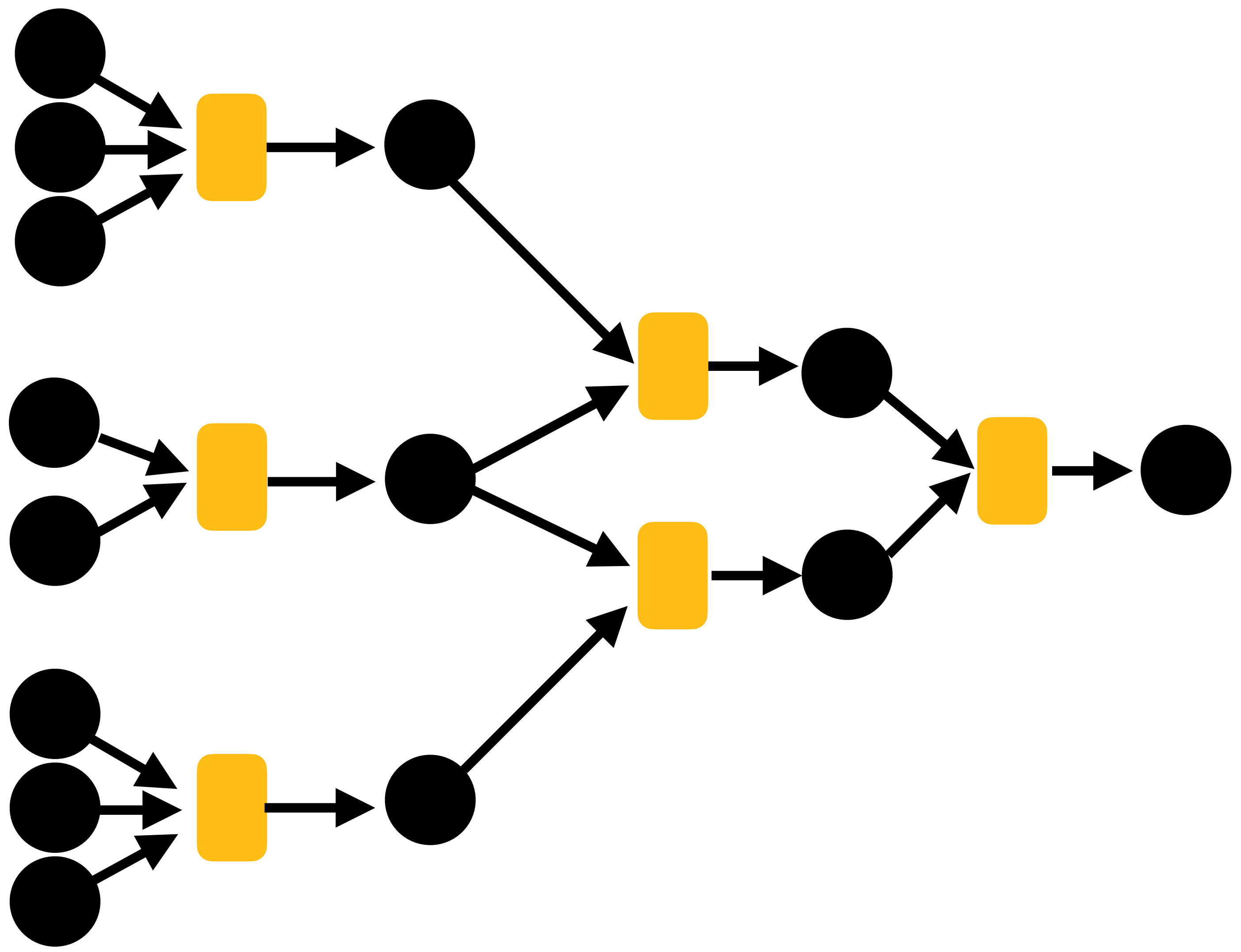
The following formula is in C^2 and cannot by MPNNs without global readout :

$$\gamma(x) = Red(x) \wedge \exists y\Big(\neg E(x,y) \wedge \exists^{\geq 2} x\big(E(y,x) \wedge Blue(x)\big)\Big),$$

since, e.g., the red and blue nodes may be in disjoint subgraphs and never communicate.

A Logical Characterization for MPNNs

$$\Phi(x) \longrightarrow$$



The proof shows how to simulate a C^2 sentence with MPNNs following the roadmap:

- Enumerate all sub-formulas $(\phi_1, \ldots, \phi_L)$ of a given formula Φ, such that $\Phi = \phi_L$
- Define an MPNN M_Φ with feature vectors in $\mathbb{R}^L$ such that every component of those vectors represents a different sub-formula.
- M_Φ updates the feature vector $\mathbf{x}_u$ of node u ensuring that its component corresponding to the sub-formula ϕ_i gets a value 1 if and only if the sub-formula ϕ_i is satisfied in node u.

Summary

- Model representation capacity & expressive power
- Graph isomorphism, color refinement, 1-WL
- MPNNs with injective aggregation and combine functions are as powerful as 1-WL test.
- The logic of graphs: FO, C, FO^2, C^2 — an interesting connection to descriptive complexity!
- Logical characterization of MPNNs
  - Each C^2 classifier can be captured by an MPNNs with global readout (even with a final readout).
  - MPNNs without global readout cannot capture C^2, but can capture graded model logic.
- We have not discussed the practical implications of the limitations in expressive power, and neither the proposed tools to address such limitations — Lecture 6 & 7.

References

- C. Morris, M. Ritzert, M. Fey, W. Hamilton,J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. *AAAI*, 2019.

- K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *ICLR*, 2019.

- P. Barcelo, E. Kostylev, M. Monet, J. Perez, J. Reutter, and J. Silva. The logical expressiveness of graph neural networks. *ICLR*, 2020.

- G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

- K. Hornik, M. Stinchcombe, H. White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989

- Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989

- L. Babai, Graph isomorphism in quasipolynomial time, arXiv:1512.03547, 2016.

- N. Immermann, Descriptive Complexity. 1999.

- J. Cai, M. Furer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. Combinatorica, 12(4):389–410, 1992.