Big Ramsey degrees of homogeneous structures
part 3: current developments and open problems

Jan Hubička

Department of Applied Mathematics
Charles University
Prague

Winter school 2022, Hejnice

Big picture: known big Ramsey results by proof techniques

Ramsey's Theorem

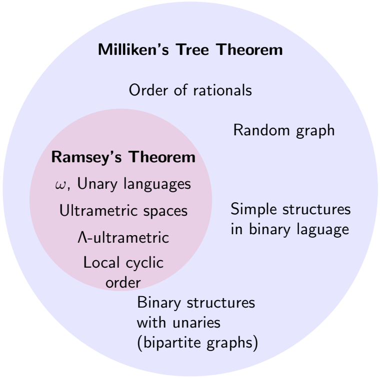
ω, Unary languages

Ultrametric spaces

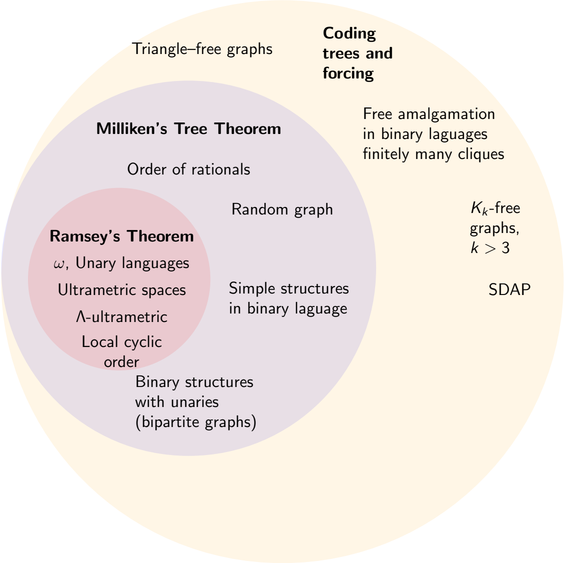
Λ-ultrametric

Local cyclic order

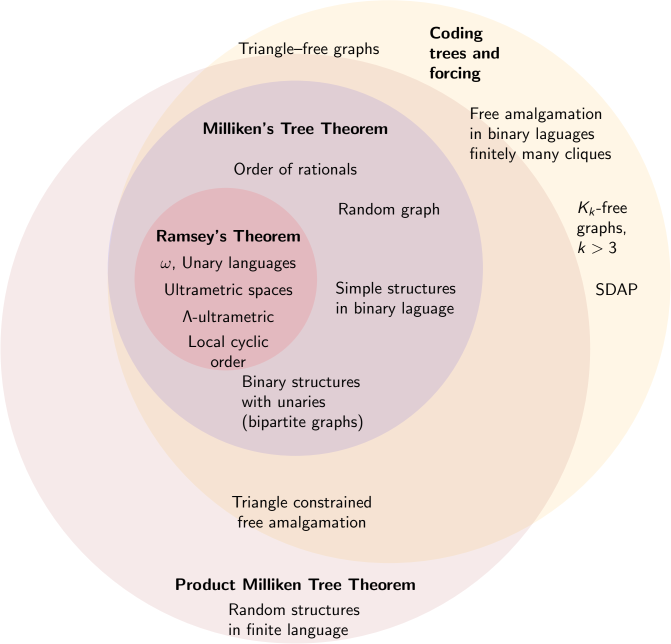
Big picture: known big Ramsey results by proof techniques



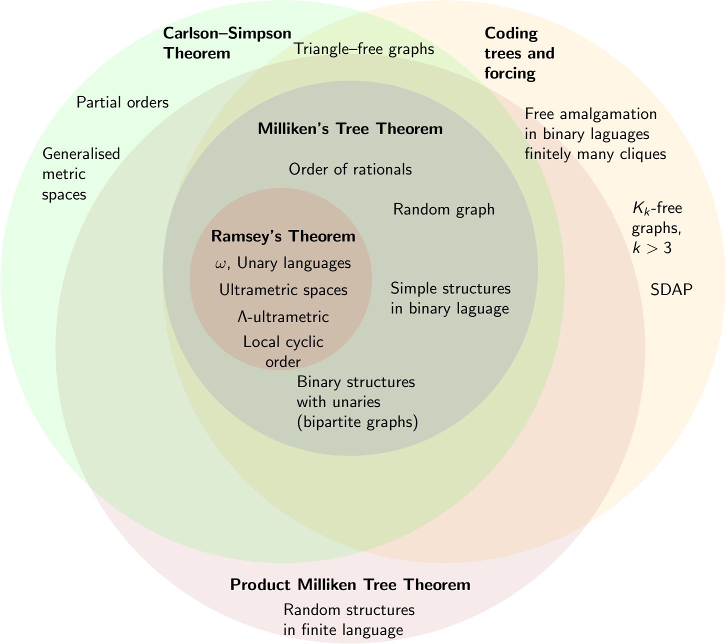
Big picture: known big Ramsey results by proof techniques



Big picture: known big Ramsey results by proof techniques



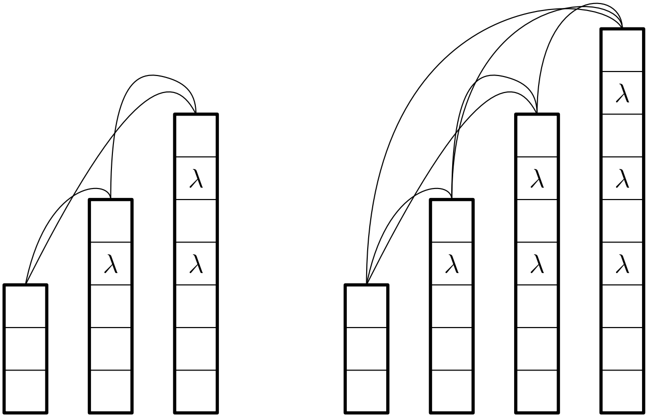
Big picture: known big Ramsey results by proof techniques



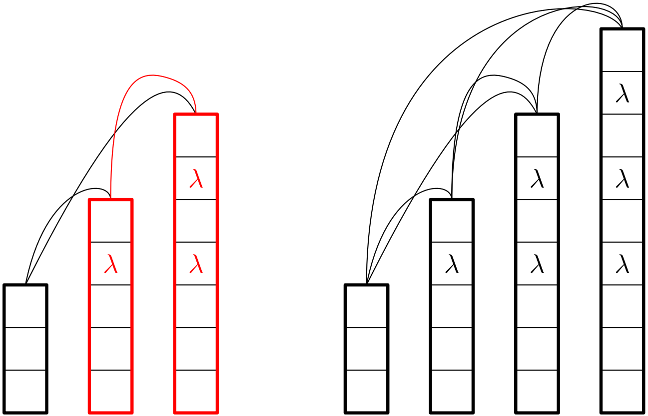
Can we find one theorem to rule them all?


S.M. Prokudin-Gorsky: Alim Khan, emir of Bukhara, 1911

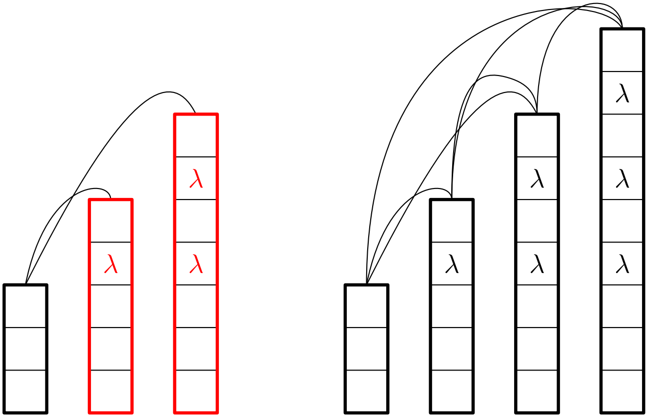
Why parameter words are not good for K_4-free graphs?



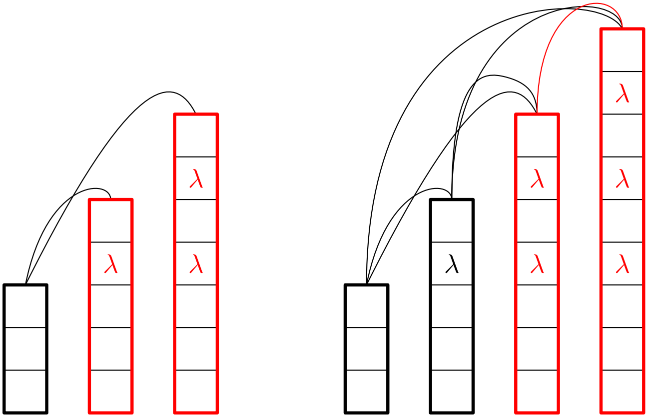
Why parameter words are not good for K_4-free graphs?



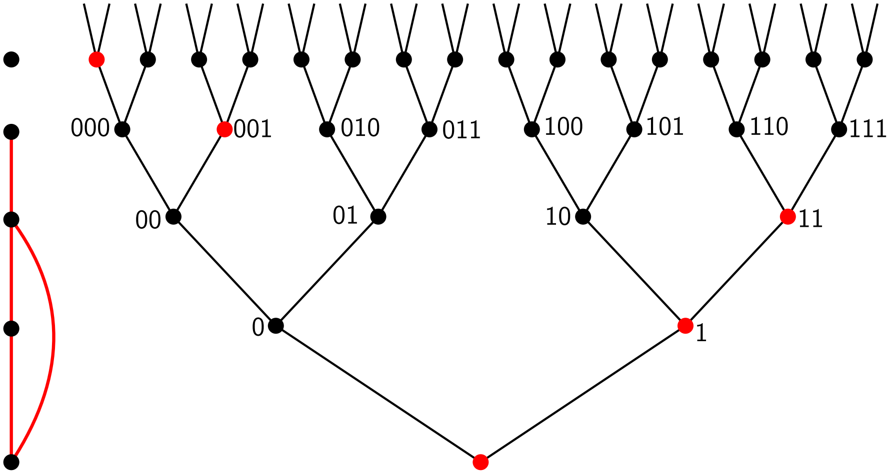
Why parameter words are not good for K_4-free graphs?



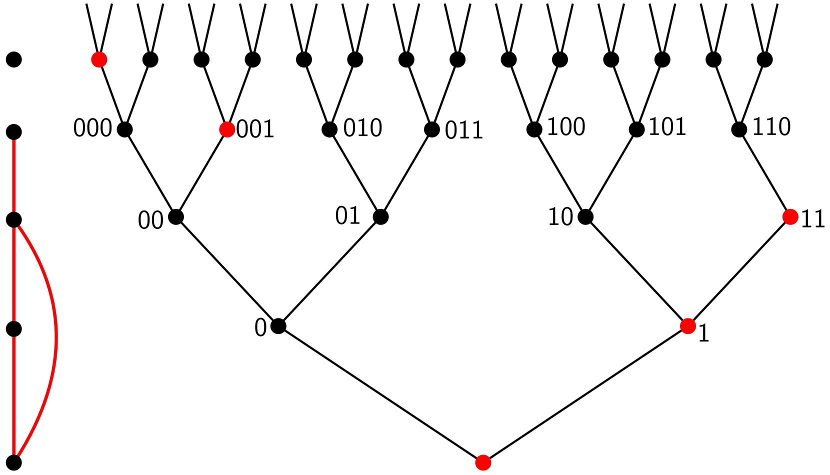
Why parameter words are not good for K_4-free graphs?



Coding tree (Dobrinen, Zucker)

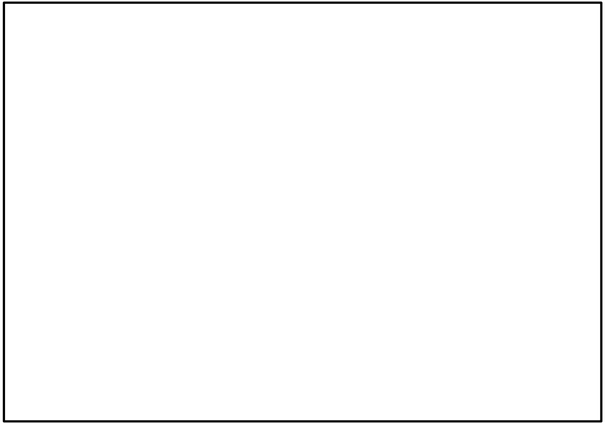


Coding tree (Dobrinen, Zucker)

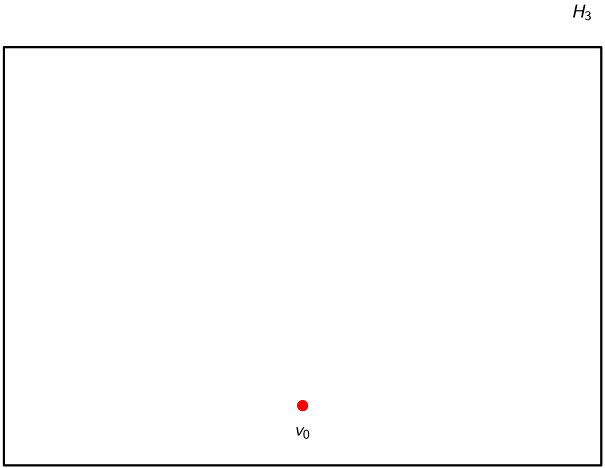


Why Laver's argument is not good for hypergraphs?

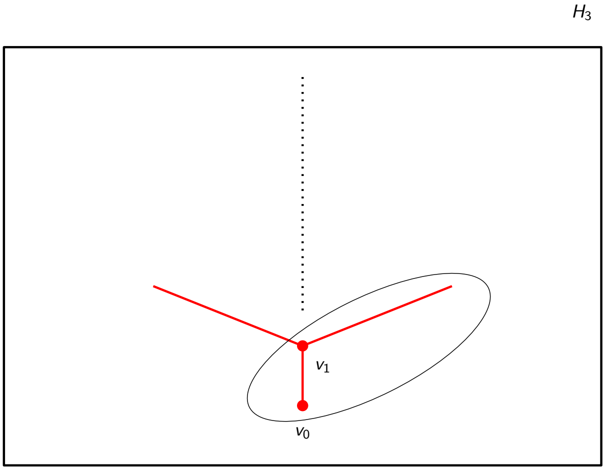
H_3



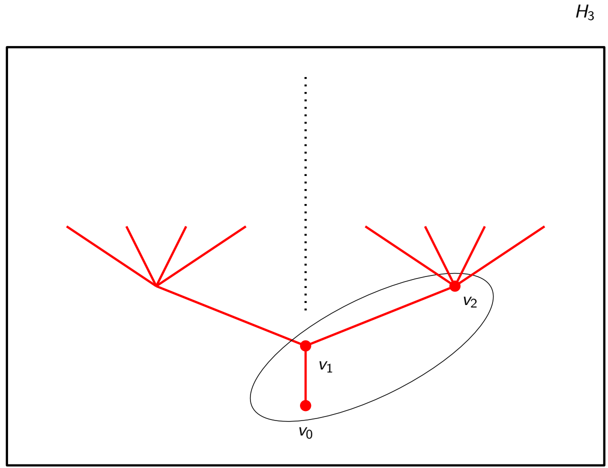
Why Laver's argument is not good for hypergraphs?



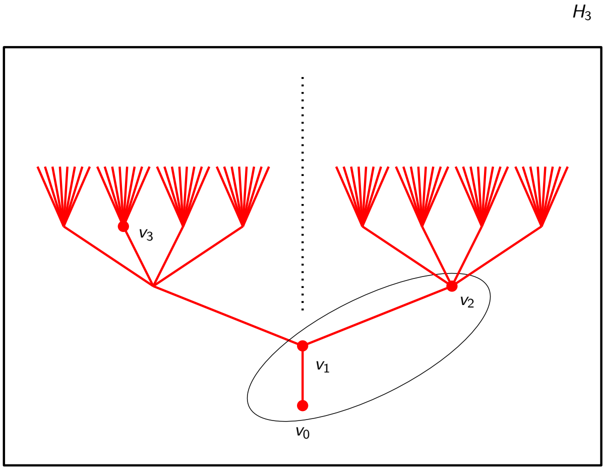
Why Laver's argument is not good for hypergraphs?



Why Laver's argument is not good for hypergraphs?

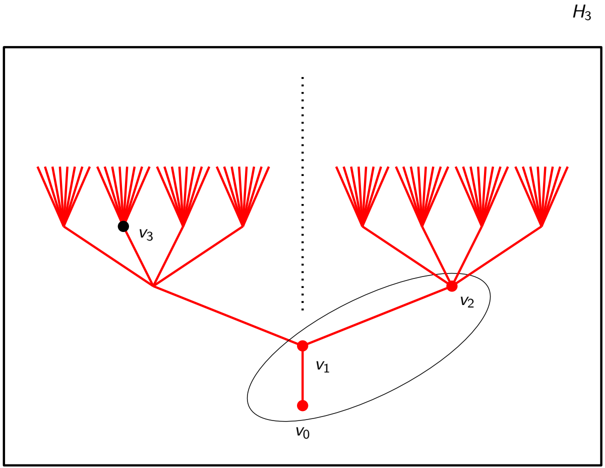


Why Laver's argument is not good for hypergraphs?



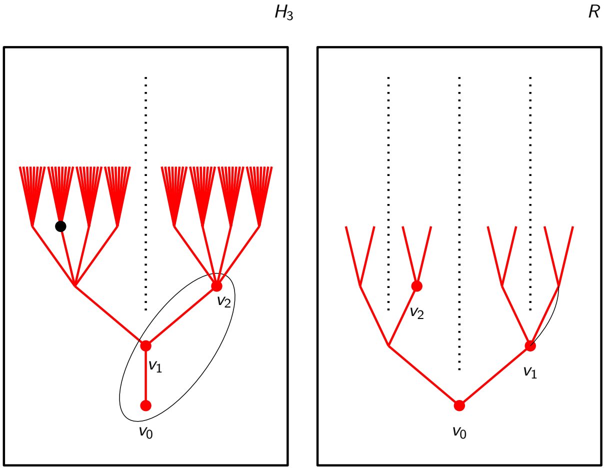
Colour of a subgraph = shape of meet closure in the tree
Problem: Ramsey theorem for this type of tree does not hold

Why Laver's argument is not good for hypergraphs?



Colour of a subgraph = shape of meet closure in the tree
Problem: Ramsey theorem for this type of tree does not hold
Year later we observed that neighbourhood of a vertex is the Random graph!

Why Laver's argument is not good for hypergraphs?



Colour of a subgraph = shape of meet closure in both trees
Problem: Ramsey theorem for this type of tree does not hold
Year later we observed that neighbourhood of a vertex is the Random graph!

“Unrestricted” structures

Theorem (Balko, Chodounsky, Jan Hubika, Konečný, Vena, 2022)

Big Ramsey degrees of the universal 3-uniform hypergraph are finite.

Theorem (Braunfeld, Chodounsky, de Rancourt, Jan Hubika, Kawach, Konečný, 2022+)

Big Ramsey degrees of the universal hypergraph are finite.

Theorem (Braunfeld, Chodounsky, de Rancourt, Jan Hubika, Kawach, Konečný, 2022+)

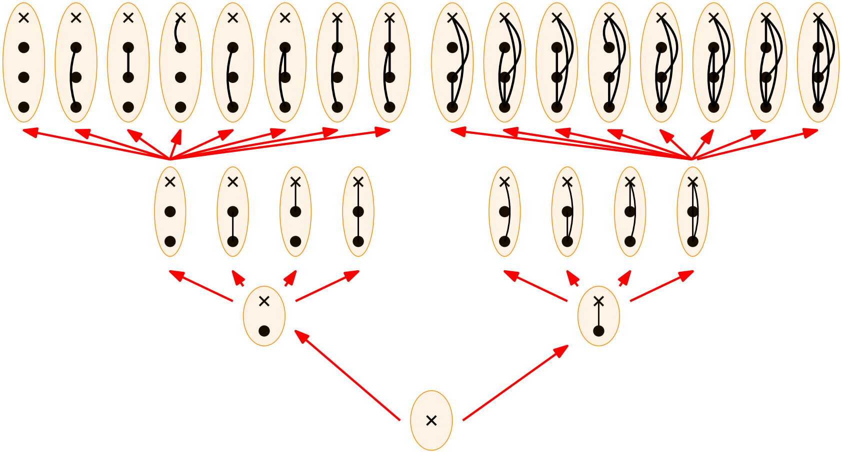
Let L be a relational language. Let $\mathcal{M}$ be a Fraïssé limit of a free amalgamation class defined by a set of forbidden structures $\mathcal{F}$. Assume that:

1. *for every $\mathbf{F} \in \mathcal{F}$ there exists $R \in L$ and $\vec{x} \in R_{\mathbf{F}}$ containing all vertices of $\mathbf{F}$, and*
2. *$\mathcal{M}$ is ω-categorical.*

Then $\mathcal{M}$ has finite big Ramsey degrees.

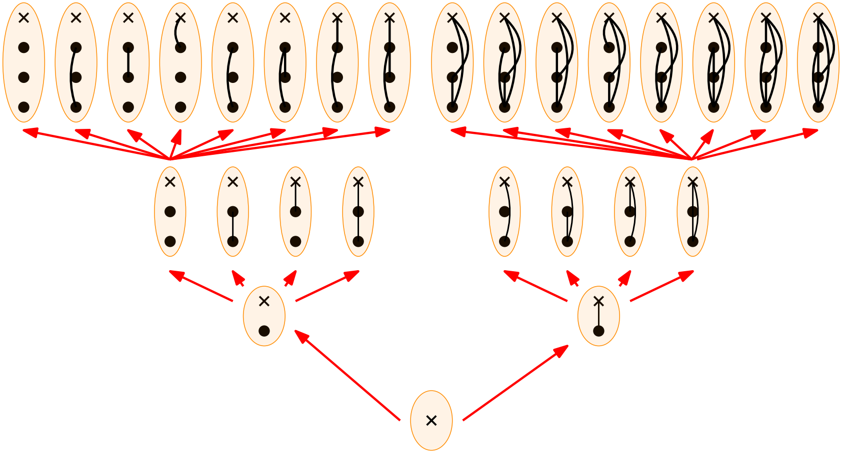
1. All results makes use of the product (or vector) form of the Milliken tree theorem.
2. Lower bounds are currently work in progress.
3. We know that the
4. The results can be extended by interposing linear orders and unary functions.

All enumerations tree



Basic idea: produce an amalgamation of all tree of types of enumerations of K_4-free graphs and make type remember the initial segment of enumeration it belongs to.

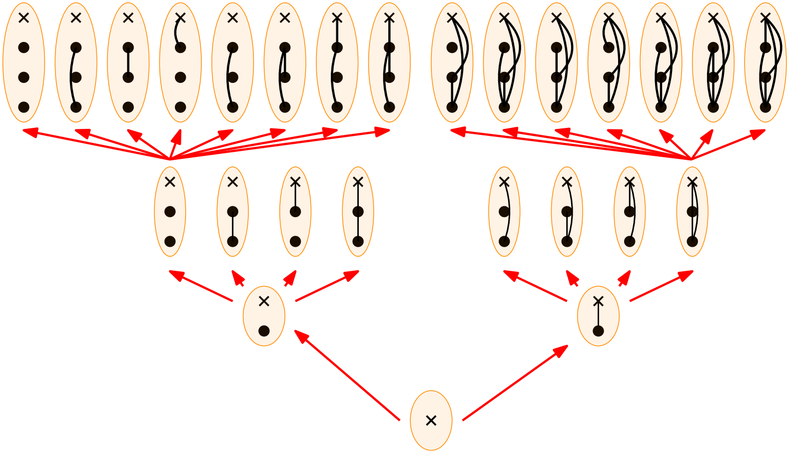
All enumerations tree



Basic idea: produce an amalgamation of all tree of types of enumerations of K_4-free graphs and make type remember the initial segment of enumeration it belongs to.

1. Type is an K_4-free graph on vertex set $\{0, 1, \ldots, n-1, t\}$. t is a type vertex denoted by cross.
2. Order is inclusion.

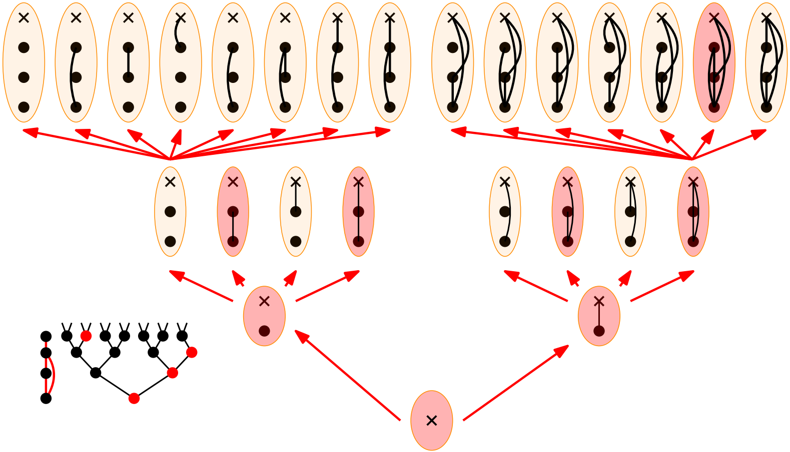
All enumerations tree



Basic idea: produce an amalgamation of all tree of types of enumerations of K_4-free graphs and make type remember the initial segment of enumeration it belongs to.

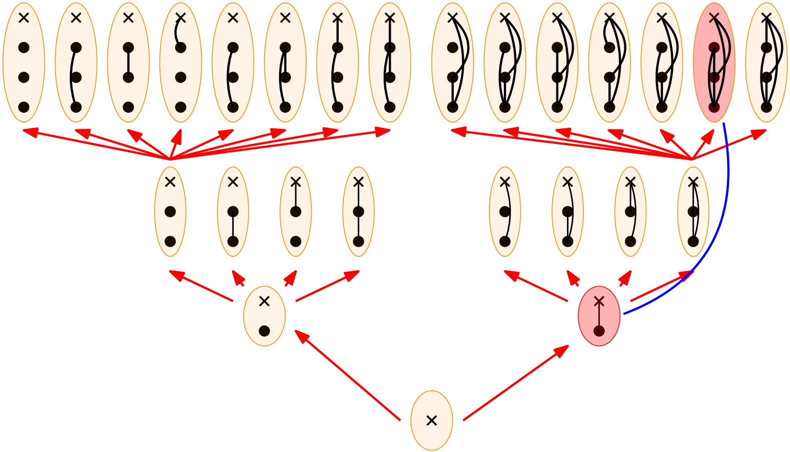
1. Type is an K_4-free graph on vertex set $\{0, 1, \ldots, n-1, t\}$. t is a type vertex denoted by cross.
2. Order is inclusion.

All enumerations tree



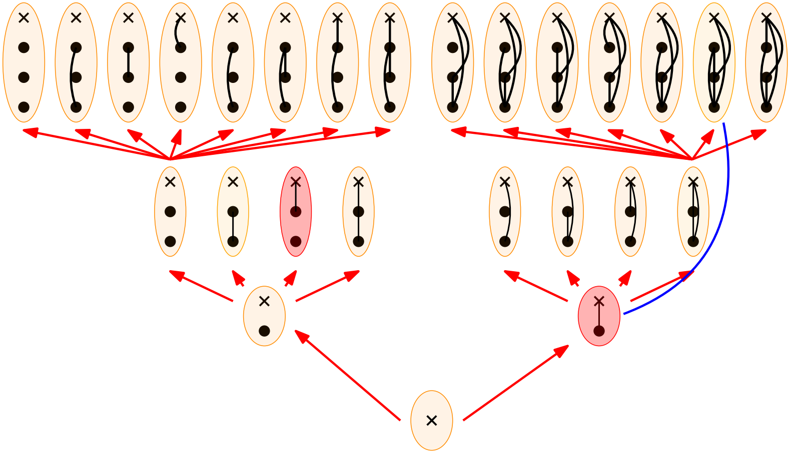
1. Every enumeration is a subtree.

All enumerations tree



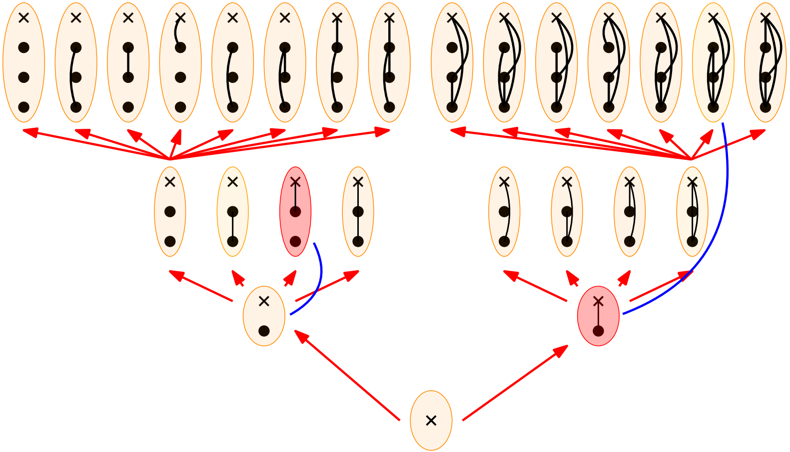
1. Every enumeration is a subtree.
2. K_4-free graph can be defined naturally on top of this tree.

All enumerations tree



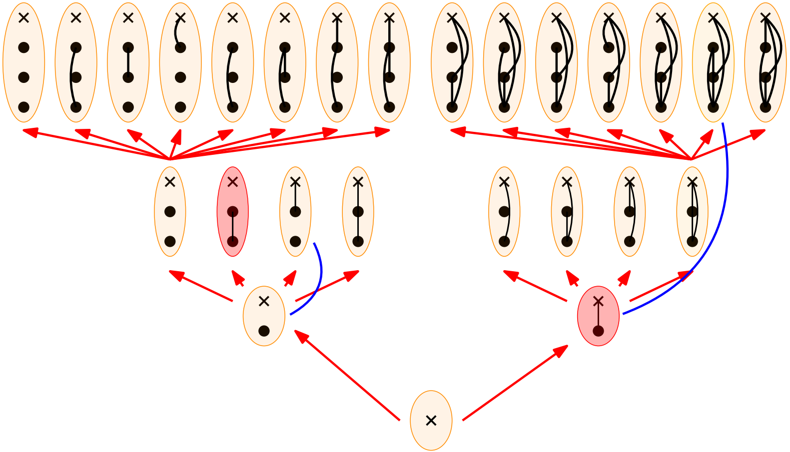
1. Every enumeration is a subtree.
2. K_4-free graph can be defined naturally on top of this tree.

All enumerations tree



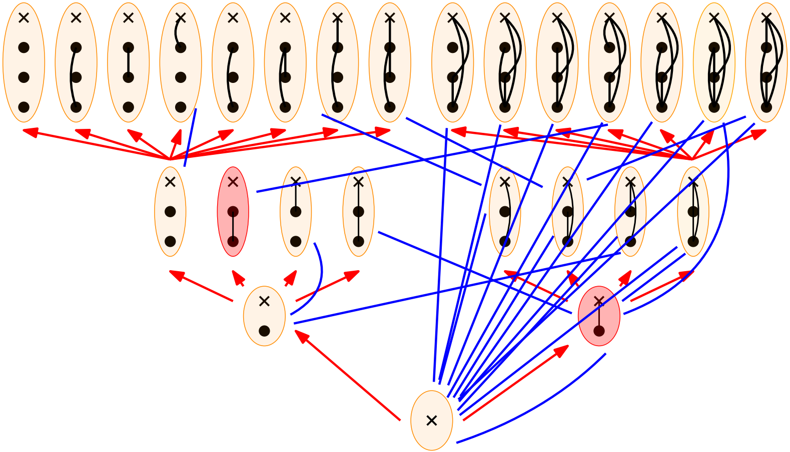
1. Every enumeration is a subtree.
2. K_4-free graph can be defined naturally on top of this tree.

All enumerations tree



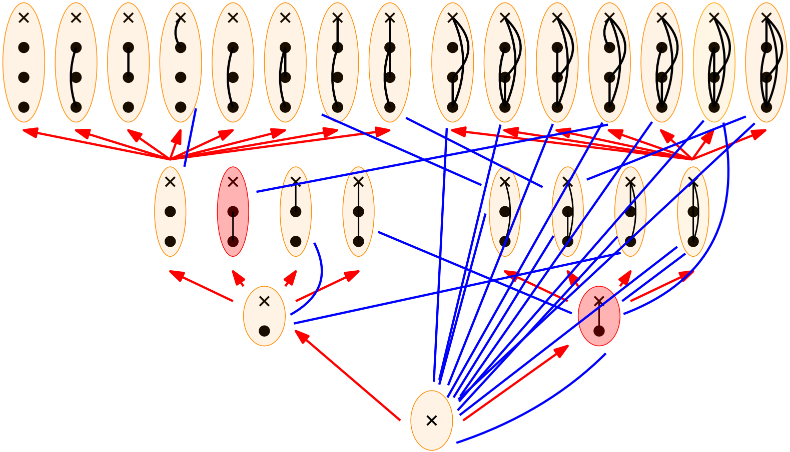
1. Every enumeration is a subtree.
2. K_4-free graph can be defined naturally on top of this tree.

All enumerations tree



1. Every enumeration is a subtree.
2. K_4-free graph can be defined naturally on top of this tree.

All enumerations tree



1. Every enumeration is a subtree.
2. K_4-free graph can be defined naturally on top of this tree.
3. Can we find a good Ramsey theorem for trees like this?

$\mathcal{S}$-trees

A tree is a (possibly empty) partially ordered set $(T, \preceq)$ such that, for every $a \in T$, the set $\{b \in T : b \prec a\}$ is finite and linearly ordered by $\preceq$.
We denote by $\ell(a)$ the level of a and by $a|_n$ the predecessor of a at level n.

Definition ($\mathcal{S}$-tree)

An $\mathcal{S}$-tree is a quadruple $(T, \preceq, \Sigma, \mathcal{S})$ where $(T, \preceq)$ is a countable finitely branching tree with finitely many nodes of level 0, Σ is a set called the alphabet and $\mathcal{S}$ is a partial function $\mathcal{S}\colon T \times T^{<\omega} \times \Sigma \to T$ called the successor operation satisfying the following three axioms:

$\mathcal{S}$-trees

A tree is a (possibly empty) partially ordered set $(T, \preceq)$ such that, for every $a \in T$, the set $\{b \in T : b \prec a\}$ is finite and linearly ordered by $\preceq$.
We denote by $\ell(a)$ the level of a and by $a|_n$ the predecessor of a at level n.

Definition ($\mathcal{S}$-tree)

An $\mathcal{S}$-tree is a quadruple $(T, \preceq, \Sigma, \mathcal{S})$ where $(T, \preceq)$ is a countable finitely branching tree with finitely many nodes of level 0, Σ is a set called the alphabet and $\mathcal{S}$ is a partial function $\mathcal{S} \colon T \times T^{<\omega} \times \Sigma \to T$ called the successor operation satisfying the following three axioms:

S1 If $\mathcal{S}(a, \bar{p}, c)$ is defined for some $a \in T$, $\bar{p} \in T^{<\omega}$ and $c \in \Sigma$, then $\mathcal{S}(a, \bar{p}, c)$ is an immediate successor of a and all nodes in $\bar{p}$ have levels at most $\ell(a) - 1$.

$\mathcal{S}$-trees

A tree is a (possibly empty) partially ordered set $(T, \preceq)$ such that, for every $a \in T$, the set $\{b \in T : b \prec a\}$ is finite and linearly ordered by $\preceq$.
We denote by $\ell(a)$ the level of a and by $a|_n$ the predecessor of a at level n.

Definition ($\mathcal{S}$-tree)

An $\mathcal{S}$-tree is a quadruple $(T, \preceq, \Sigma, \mathcal{S})$ where $(T, \preceq)$ is a countable finitely branching tree with finitely many nodes of level 0, Σ is a set called the alphabet and $\mathcal{S}$ is a partial function $\mathcal{S}\colon T \times T^{<\omega} \times \Sigma \to T$ called the successor operation satisfying the following three axioms:

- S1 If $\mathcal{S}(a, \bar{p}, c)$ is defined for some $a \in T$, $\bar{p} \in T^{<\omega}$ and $c \in \Sigma$, then $\mathcal{S}(a, \bar{p}, c)$ is an immediate successor of a and all nodes in $\bar{p}$ have levels at most $\ell(a) - 1$.
- S2 Injectivity: If $\mathcal{S}(a, \bar{p}, c) = \mathcal{S}(b, \bar{q}, d)$, then $a = b$, $\bar{p} = \bar{q}$ and $c = d$.

$\mathcal{S}$-trees

A tree is a (possibly empty) partially ordered set $(T, \preceq)$ such that, for every $a \in T$, the set $\{b \in T : b \prec a\}$ is finite and linearly ordered by $\preceq$.
We denote by $\ell(a)$ the level of a and by $a|_n$ the predecessor of a at level n.

Definition ($\mathcal{S}$-tree)

An $\mathcal{S}$-tree is a quadruple $(T, \preceq, \Sigma, \mathcal{S})$ where $(T, \preceq)$ is a countable finitely branching tree with finitely many nodes of level 0, Σ is a set called the alphabet and $\mathcal{S}$ is a partial function $\mathcal{S}\colon T \times T^{<\omega} \times \Sigma \to T$ called the successor operation satisfying the following three axioms:

- **S1** If $\mathcal{S}(a, \bar{p}, c)$ is defined for some $a \in T$, $\bar{p} \in T^{<\omega}$ and $c \in \Sigma$, then $\mathcal{S}(a, \bar{p}, c)$ is an immediate successor of a and all nodes in $\bar{p}$ have levels at most $\ell(a) - 1$.
- **S2** Injectivity: If $\mathcal{S}(a, \bar{p}, c) = \mathcal{S}(b, \bar{q}, d)$, then $a = b$, $\bar{p} = \bar{q}$ and $c = d$.
- **S3** Constructivity: For every node $a \in T$ of level at least 1, there exist $\bar{p} \in T^{<\omega}$ and $c \in \Sigma$ such that $\mathcal{S}(a|_{\ell(a)-1}, \bar{p}, c) = a$.

$\mathcal{S}$-trees

A tree is a (possibly empty) partially ordered set $(T, \preceq)$ such that, for every $a \in T$, the set $\{b \in T : b \prec a\}$ is finite and linearly ordered by $\preceq$.
We denote by $\ell(a)$ the level of a and by $a|_n$ the predecessor of a at level n.

Definition ($\mathcal{S}$-tree)

An $\mathcal{S}$-tree is a quadruple $(T, \preceq, \Sigma, \mathcal{S})$ where $(T, \preceq)$ is a countable finitely branching tree with finitely many nodes of level 0, Σ is a set called the alphabet and $\mathcal{S}$ is a partial function $\mathcal{S}\colon T \times T^{<\omega} \times \Sigma \to T$ called the successor operation satisfying the following three axioms:

- S1 If $\mathcal{S}(a, \bar{p}, c)$ is defined for some $a \in T$, $\bar{p} \in T^{<\omega}$ and $c \in \Sigma$, then $\mathcal{S}(a, \bar{p}, c)$ is an immediate successor of a and all nodes in $\bar{p}$ have levels at most $\ell(a) - 1$.
- S2 Injectivity: If $\mathcal{S}(a, \bar{p}, c) = \mathcal{S}(b, \bar{q}, d)$, then $a = b$, $\bar{p} = \bar{q}$ and $c = d$.
- S3 Constructivity: For every node $a \in T$ of level at least 1, there exist $\bar{p} \in T^{<\omega}$ and $c \in \Sigma$ such that $\mathcal{S}(a|_{\ell(a)-1}, \bar{p}, c) = a$.

Example

Consider the binary tree of $\{0, 1\}$-words $(B, \sqsubseteq)$ and denote by r its root. $\mathcal{S}$ can be defined only for empty $\bar{p}$ as a concatenation.

$$01011 = \mathcal{S}(\mathcal{S}(\mathcal{S}(\mathcal{S}(\mathcal{S}(r, (), 0), (), 1), (), 0), (), 1), (), 1).$$

Level-decomposition

Definition ($\mathcal{S}$-term)

Given an $\mathcal{S}$-tree $(T, \preceq, \Sigma, \mathcal{S})$, we call a term α an $\mathcal{S}$-term if and only if $\alpha \in T$, or $\alpha = (\beta, (\gamma_0, \gamma_1, \ldots, \gamma_{n-1}), c)$ where $n \in \omega$, all of $\beta, \gamma_0, \gamma_1 \ldots \gamma_{n-1}$ are $\mathcal{S}$-terms and $c \in \Sigma$.

Definition (Level decomposition)

Let $(T, \preceq, \Sigma, \mathcal{S})$ be an $\mathcal{S}$-tree. Given $a \in T$ and $n < \omega$, the level n decomposition of a, denoted by $\mathcal{D}_n(a)$, is an $\mathcal{S}$-term defined recursively:

1. If $\ell(a) \leq n$, then
$$\mathcal{D}_n(a) = a.$$
2. For $a = \mathcal{S}(b, (p_0, \ldots, p_{n-1}), c)$ such that $\ell(a) > n$, we let
$$\mathcal{D}_n(a) = (\mathcal{D}_n(b), (\mathcal{D}_n(p_0), \mathcal{D}_n(p_1), \ldots, \mathcal{D}_n(p_{n-1})), c).$$

Level-decomposition

Definition ($\mathcal{S}$-term)

Given an $\mathcal{S}$-tree $(T, \preceq, \Sigma, \mathcal{S})$, we call a term α an $\mathcal{S}$-term if and only if $\alpha \in T$, or $\alpha = (\beta, (\gamma_0, \gamma_1, \ldots, \gamma_{n-1}), c)$ where $n \in \omega$, all of $\beta, \gamma_0, \gamma_1 \ldots \gamma_{n-1}$ are $\mathcal{S}$-terms and $c \in \Sigma$.

Definition (Level decomposition)

Let $(T, \preceq, \Sigma, \mathcal{S})$ be an $\mathcal{S}$-tree. Given $a \in T$ and $n < \omega$, the level n decomposition of a, denoted by $\mathcal{D}_n(a)$, is an $\mathcal{S}$-term defined recursively:

1. If $\ell(a) \leq n$, then
$$\mathcal{D}_n(a) = a.$$
2. For $a = \mathcal{S}(b, (p_0, \ldots, p_{n-1}), c)$ such that $\ell(a) > n$, we let
$$\mathcal{D}_n(a) = (\mathcal{D}_n(b), (\mathcal{D}_n(p_0), \mathcal{D}_n(p_1), \ldots, \mathcal{D}_n(p_{n-1})), c).$$

Example

$$\mathcal{D}_1(001) = ((0, (), 0), (), 1).$$

Manipulating nodes

We denote the class of all $\mathcal{S}$-terms by $\mathcal{T}$. For a set $S \subseteq T$ and a function $f\colon S \to \mathcal{T}$, we denote by $f(\alpha)$ the $\mathcal{S}$-term defined recursively as:

$$f(\alpha) = \begin{cases} f(\alpha) & \text{if } \alpha \in S, \\ \alpha & \text{if } \alpha \in T \setminus S, \\ (f(\beta), (f(\gamma_0), f(\gamma_1), \ldots, f(\gamma_{n-1})), c) & \text{if } \alpha = (\beta, (\gamma_0, \gamma_1, \ldots, \gamma_{n-1}), c). \end{cases}$$

Manipulating nodes

We denote the class of all $\mathcal{S}$-terms by $\mathcal{T}$. For a set $S \subseteq T$ and a function $f\colon S \to \mathcal{T}$, we denote by $f(\alpha)$ the $\mathcal{S}$-term defined recursively as:

$$f(\alpha) = \begin{cases} f(\alpha) & \text{if } \alpha \in S, \\ \alpha & \text{if } \alpha \in T \setminus S, \\ (f(\beta), (f(\gamma_0), f(\gamma_1), \ldots, f(\gamma_{n-1})), c) & \text{if } \alpha = (\beta, (\gamma_0, \gamma_1, \ldots, \gamma_{n-1}), c). \end{cases}$$

Definition (Level removal)

Given $a \in T$ and $n < \ell(a)$, we let $R_n(a)$ be a node $b \in T$ satisfying $\mathcal{D}_n(b) = r_n(\mathcal{D}_{n+1}(a))$ where r_n is a function $r_n\colon T(n+1) \to \mathcal{T}$ defined by $r_n(d) = d|_n$. If there is no such node b, we say that $R_n(a)$ is undefined.

Manipulating nodes

We denote the class of all $\mathcal{S}$-terms by $\mathcal{T}$. For a set $S \subseteq T$ and a function $f \colon S \to \mathcal{T}$, we denote by $f(\alpha)$ the $\mathcal{S}$-term defined recursively as:

$$f(\alpha) = \begin{cases} f(\alpha) & \text{if } \alpha \in S, \\ \alpha & \text{if } \alpha \in T \setminus S, \\ (f(\beta), (f(\gamma_0), f(\gamma_1), \ldots, f(\gamma_{n-1})), c) & \text{if } \alpha = (\beta, (\gamma_0, \gamma_1, \ldots, \gamma_{n-1}), c). \end{cases}$$

Definition (Level removal)

Given $a \in T$ and $n < \ell(a)$, we let $R_n(a)$ be a node $b \in T$ satisfying $\mathcal{D}_n(b) = r_n(\mathcal{D}_{n+1}(a))$ where r_n is a function $r_n \colon T(n+1) \to \mathcal{T}$ defined by $r_n(d) = d|_n$. If there is no such node b, we say that $R_n(a)$ is undefined.

Definition (Level duplication)

Given $a \in T$ and $m < n \leq \ell(a)$, we let $C^n_m(a)$ be a node $b \in T$ satisfying $\mathcal{D}_n(b) = c^n_m(\mathcal{D}_n(a))$ where c^n_m is a function $c^n_m \colon T(n) \to \mathcal{T}$ defined by $c^n_m(d) = (d, \bar{p}, c)$ where $d|_{m+1} = \mathcal{S}(d_m, \bar{p}, c)$. If there is no such node b, we say that $C^n_m(a)$ is undefined.

Definition (Shape-preserving functions)

Let $(T, \preceq, \Sigma, \mathcal{S})$ be an $\mathcal{S}$-tree. We call a function $F\colon T \to T$ a shape-preserving function if

1. F is level preserving, and
2. F is weakly $\mathcal{S}$-preserving: If $a = \mathcal{S}(b, \bar{p}, c)$ then $F(a) \preceq \mathcal{S}(F(b), F(\bar{p}), c)$

Function $f\colon S \to T$, $S \subseteq T$ is shape-preserving if it extends to a shape-pres. $F\colon T \to T$.

Definition (Shape-preserving functions)

Let $(T, \preceq, \Sigma, \mathcal{S})$ be an $\mathcal{S}$-tree. We call a function $F\colon T \to T$ a shape-preserving function if

1. F is level preserving, and
2. F is weakly $\mathcal{S}$-preserving: If $a = \mathcal{S}(b, \bar{p}, c)$ then $F(a) \preceq \mathcal{S}(F(b), F(\bar{p}), c)$

Function $f\colon S \to T$, $S \subseteq T$ is shape-preserving if it extends to a shape-pres. $F\colon T \to T$.

Shape(S, S') is the set all shape-preserving functions $f : S \to T$, $f[S] \subseteq S'$.

Definition (Shape-preserving functions)

Let $(T, \preceq, \Sigma, \mathcal{S})$ be an $\mathcal{S}$-tree. We call a function $F\colon T \to T$ a shape-preserving function if

1. F is level preserving, and
2. F is weakly $\mathcal{S}$-preserving: If $a = \mathcal{S}(b, \bar{p}, c)$ then $F(a) \preceq \mathcal{S}(F(b), F(\bar{p}), c)$

Function $f\colon S \to T$, $S \subseteq T$ is shape-preserving if it extends to a shape-pres. $F\colon T \to T$.

$\mathrm{Shape}(S, S')$ is the set all shape-preserving functions $f : S \to T$, $f[S] \subseteq S'$.

Theorem (Balko, Chodounský, Dobrinen, H., Konečný, Nešetřil, Zucker, Vena, 2021+)

Let $(T, \preceq, \Sigma, \mathcal{S})$ be an $\mathcal{S}$-tree. Assume that $\mathcal{S}$ satisfies the following conditions:

- S4 *Level removal: For every $a \in T$, $n < \ell(a)$ such that $\mathcal{D}_{n+1}(a)$ does not use any nodes of level n, the node $R_n(a)$ is defined.*
- S5 *Level duplication: For every $a \in T$, $m < n \leq \ell(a)$, the node $C^n_m(a)$ is defined.*
- S6 *Decomposition: For every $n \in \omega$, $g \in \mathrm{Shape}(T(\leq n), T)$ such that $n > 0$ and $\tilde{g}(n) > \tilde{g}(n-1) + 1$, there exists $g_1 \in \mathrm{Shape}(T(\leq n), T)$ and $g_2 \in \mathrm{Shape}_{\tilde{g}(n)-1}(T(\leq(\tilde{g}(n) - 1), T))$ such that $\tilde{g}_1(n) = \tilde{g}(n) - 1$ and $g_2 \circ g_1 = g$.*

Then, for every $k \in \omega$ and every finite colouring χ of $\mathrm{Shape}(T(\leq k), T)$, there exists $F \in \mathrm{Shape}(T, T)$ such that χ is constant when restricted to $\mathrm{Shape}(T(\leq k), F[T])$.

Ramsey theorem for shape-preserving functions

Theorem (Balko, Chodounský, Dobrinen, H., Konečný, Nešetřil, Zucker, Vena, 2021+)

Let $(T, \preceq, \Sigma, \mathcal{S})$ be an $\mathcal{S}$-tree. Assume that $\mathcal{S}$ satisfies the following conditions:

- S4 *Level removal: For every $a \in T, n < \ell(a)$ such that $\mathcal{D}_{n+1}(a)$ does not use any nodes of level n, the node $R_n(a)$ is defined.*
- S5 *Level duplication: For every $a \in T, m < n \leq \ell(a)$, the node $C^n_m(a)$ is defined.*
- S6 *Decomposition: For every $n \in \omega, g \in \mathrm{Shape}(T(\leq n), T)$ such that $n > 0$ and $\tilde{g}(n) > \tilde{g}(n-1) + 1$, there exists $g_1 \in \mathrm{Shape}(T(\leq n), T)$ and $g_2 \in \mathrm{Shape}_{\tilde{g}(n)-1}(T(\leq(\tilde{g}(n) - 1), T))$ such that $\tilde{g}_1(n) = \tilde{g}(n) - 1$ and $g_2 \circ g_1 = g$.*

Then, for every $k \in \omega$ and every finite colouring χ of $\mathrm{Shape}(T(\leq k), T)$, there exists $F \in \mathrm{Shape}(T, T)$ such that χ is constant when restricted to $\mathrm{Shape}(T(\leq k), F[T])$.

Proof outline (5 pages)

1. Use Hales-Jewett theorem to prove 1-dimensional pigeonhole
2. Use combinatorial forcing to prove ω-dimensional pigeonhole
3. Use fusion like in proof of Milliken's theorem to prove the theorem

Application to K_4-free graphs

Definition (Type)

Type of level n is a K_4-free graph $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

Definition (Levelled type)

Levelled type of level n is a pair $\mathbf{a} = (\mathbf{A}, \mathrm{fl}_{\mathbf{A}})$ where $\mathbf{A}$ is a type of level n and $\mathrm{fl} : n \setminus \{0\} \to n$ is a function satisfying:

1. $\mathrm{fl}_{\mathbf{a}}(i) < i$.
2. whenever $i < j$ forms an edge of $\mathbf{A}$ then $\mathrm{fl}_{\mathbf{A}}(j) > i$.

Nodes of an $\mathcal{S}$-tree are levelled types ordered by inclusion. Successor operation is an amalgamation.

Application to K_4-free graphs

Definition (Type)

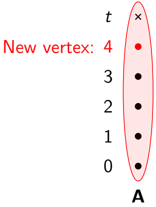
Type of level n is a K_4-free graph $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

Definition (Levelled type)

Levelled type of level n is a pair $\mathbf{a} = (\mathbf{A}, \mathrm{fl}_\mathbf{A})$ where $\mathbf{A}$ is a type of level n and $\mathrm{fl} : n \setminus \{0\} \to n$ is a function satisfying:

1. $\mathrm{fl}_\mathbf{a}(i) < i$.
2. whenever $i < j$ forms an edge of $\mathbf{A}$ then $\mathrm{fl}_\mathbf{A}(j) > i$.

Nodes of an $\mathcal{S}$-tree are levelled types ordered by inclusion. Successor operation is an amalgamation.



Application to K_4-free graphs

Definition (Type)

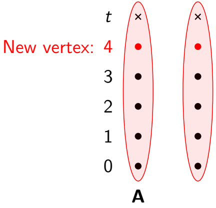
Type of level n is a K_4-free graph $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

Definition (Levelled type)

Levelled type of level n is a pair $\mathbf{a} = (\mathbf{A}, \mathrm{fl}_{\mathbf{A}})$ where $\mathbf{A}$ is a type of level n and $\mathrm{fl} : n \setminus \{0\} \to n$ is a function satisfying:

1. $\mathrm{fl}_{\mathbf{a}}(i) < i$.
2. whenever $i < j$ forms an edge of $\mathbf{A}$ then $\mathrm{fl}_{\mathbf{A}}(j) > i$.

Nodes of an $\mathcal{S}$-tree are levelled types ordered by inclusion. Successor operation is an amalgamation.



Application to K_4-free graphs

Definition (Type)

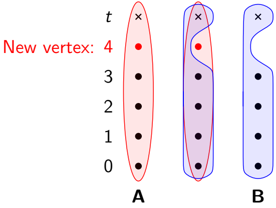
Type of level n is a K_4-free graph $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

Definition (Levelled type)

Levelled type of level n is a pair $\mathbf{a} = (\mathbf{A}, \mathrm{fl}_{\mathbf{A}})$ where $\mathbf{A}$ is a type of level n and $\mathrm{fl} : n \setminus \{0\} \to n$ is a function satisfying:

1. $\mathrm{fl}_{\mathbf{a}}(i) < i$.
2. whenever $i < j$ forms an edge of $\mathbf{A}$ then $\mathrm{fl}_{\mathbf{A}}(j) > i$.

Nodes of an $\mathcal{S}$-tree are levelled types ordered by inclusion. Successor operation is an amalgamation.



Application to K_4-free graphs

Definition (Type)

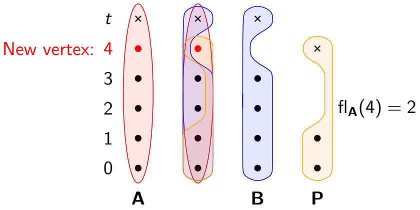
Type of level n is a K_4-free graph $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

Definition (Levelled type)

Levelled type of level n is a pair $\mathbf{a} = (\mathbf{A}, \mathrm{fl}_{\mathbf{A}})$ where $\mathbf{A}$ is a type of level n and $\mathrm{fl} : n \setminus \{0\} \to n$ is a function satisfying:

1. $\mathrm{fl}_{\mathbf{a}}(i) < i$.
2. whenever $i < j$ forms an edge of $\mathbf{A}$ then $\mathrm{fl}_{\mathbf{A}}(j) > i$.

Nodes of an $\mathcal{S}$-tree are levelled types ordered by inclusion. Successor operation is an amalgamation.



Application to K_4-free graphs

Definition (Type)

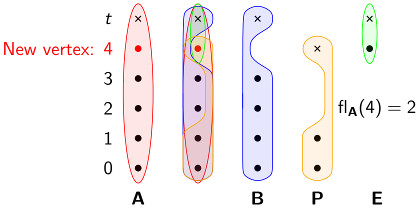
Type of level n is a K_4-free graph $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

Definition (Levelled type)

Levelled type of level n is a pair $\mathbf{a} = (\mathbf{A}, \mathrm{fl}_{\mathbf{A}})$ where $\mathbf{A}$ is a type of level n and $\mathrm{fl} : n \setminus \{0\} \to n$ is a function satisfying:

1. $\mathrm{fl}_{\mathbf{a}}(i) < i$.
2. whenever $i < j$ forms an edge of $\mathbf{A}$ then $\mathrm{fl}_{\mathbf{A}}(j) > i$.

Nodes of an $\mathcal{S}$-tree are levelled types ordered by inclusion. Successor operation is an amalgamation.



Application to K_4-free graphs

Definition (Type)

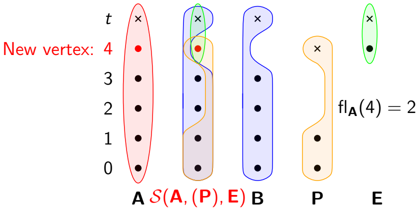
Type of level n is a K_4-free graph $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

Definition (Levelled type)

Levelled type of level n is a pair $\mathbf{a} = (\mathbf{A}, \mathrm{fl}_{\mathbf{A}})$ where $\mathbf{A}$ is a type of level n and $\mathrm{fl} : n \setminus \{0\} \to n$ is a function satisfying:

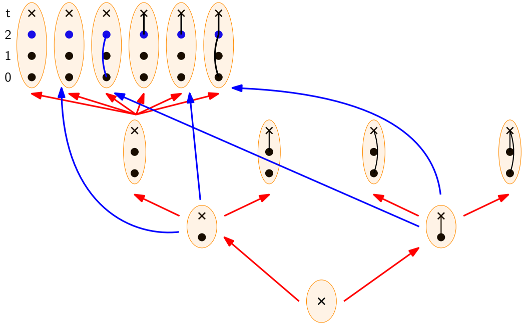
1. $\mathrm{fl}_{\mathbf{a}}(i) < i$.
2. whenever $i < j$ forms an edge of $\mathbf{A}$ then $\mathrm{fl}_{\mathbf{A}}(j) > i$.

Nodes of an $\mathcal{S}$-tree are levelled types ordered by inclusion. Successor operation is an amalgamation.



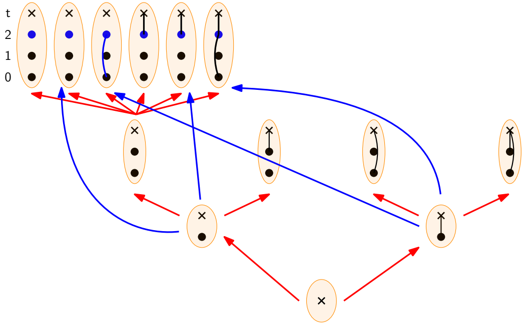
Non-forcing proof of Zucker's theorem

1. Build an $\mathcal{S}$-tree of levelled types:



Non-forcing proof of Zucker's theorem

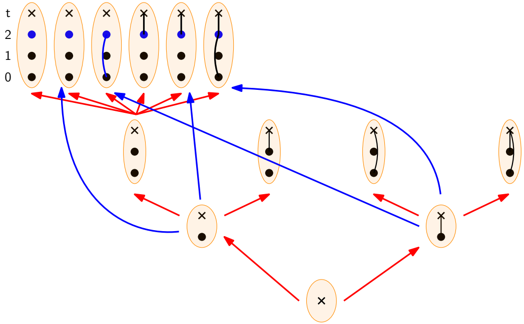
1. Build an $\mathcal{S}$-tree of levelled types:



2. Axioms S1, S2, S3 follows by construction.

Non-forcing proof of Zucker's theorem

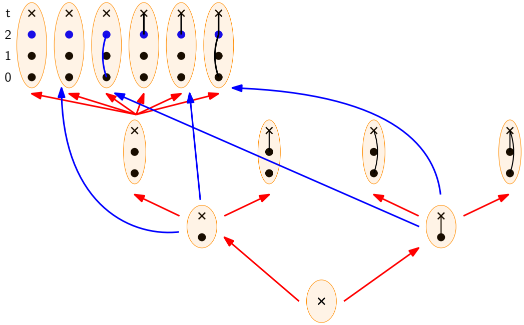
1. Build an $\mathcal{S}$-tree of levelled types:



2. Axioms S1, S2, S3 follows by construction.
3. Axiom S4 (level removal) follows from hereditarity of the types
4. Axiom S5 (level duplication) follows from free amalgamation property
5. Axiom S6 (decomposition) follows from free amalgamation property

Non-forcing proof of Zucker's theorem

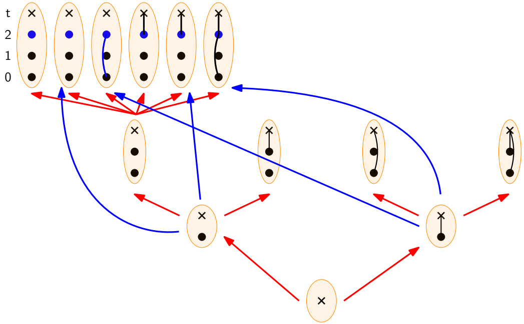
1. Build an $\mathcal{S}$-tree of levelled types:



2. Axioms S1, S2, S3 follows by construction.
3. Axiom S4 (level removal) follows from hereditarity of the types
4. Axiom S5 (level duplication) follows from free amalgamation property
5. Axiom S6 (decomposition) follows from free amalgamation property
6. Define structure on nodes of the $\mathcal{S}$-tree and verify that shape-preserving functions preserve the structure

Non-forcing proof of Zucker's theorem

1. Build an $\mathcal{S}$-tree of levelled types:



2. Axioms S1, S2, S3 follows by construction.
3. Axiom S4 (level removal) follows from hereditarity of the types
4. Axiom S5 (level duplication) follows from free amalgamation property
5. Axiom S6 (decomposition) follows from free amalgamation property
6. Define structure on nodes of the $\mathcal{S}$-tree and verify that shape-preserving functions preserve the structure
7. Verify that envelopes are bounded for nice copies inside nice enumerations (same was as in Zucker's paper)

Remarks

1. The proof generalizes naturally to strong amalgamation classes including partial orders, special metric spaces.
   (general theorem in work in progress.)
2. Optimal upper bounds on big Ramsey degrees can be achieved.
3. For languages with relations of higher arity the $\mathcal{S}$-tree can be defined analogously using n-types instead of 1-types. Surprising complication occurs when forbidding some substructures: the case of free amalgamation classes in finite languages is still open.

Open problems

1. Big Ramsey degrees for free amalgamation classes in finite binary languages with infinitely many forbidden substructures.

Open problems

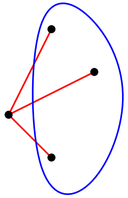
1. Big Ramsey degrees for free amalgamation classes in finite binary languages with infinitely many forbidden substructures.
2. What is the right structural condition for non-free amalgamation classes in finite binary languages for them to have finite big Ramsey degrees?

Open problems

1. Big Ramsey degrees for free amalgamation classes in finite binary languages with infinitely many forbidden substructures.
2. What is the right structural condition for non-free amalgamation classes in finite binary languages for them to have finite big Ramsey degrees?
3. What about non-unary function symbols?

Open problems

1. Big Ramsey degrees for free amalgamation classes in finite binary languages with infinitely many forbidden substructures.
2. What is the right structural condition for non-free amalgamation classes in finite binary languages for them to have finite big Ramsey degrees?
3. What about non-unary function symbols?
4. Big Ramsey degrees for free amalgamation classes in finite languages (of higher arity).
   In particular we do not know how to forbid the following:



In a Land of Fantastic Cacti, Fred Payne Clatworthy, Autochrome, 7 x 5", c1929
Mark Jacobs Collection

Optimality

Optimality

Known characterisations of big Ramsey degrees:

1. 1979 Devlin: the order of rationals
2. 2008 laflamme, Nguyen Van Th, Sauer: Ultrametric spaces
3. 2010 Laflamme, Sauer, Vuskanovic: the Rado graph
4. 2010 Balko, Chodounský, Hubička, Konečný, Vena, Zucker; independenty Dobrinen: Triangle free graphs
5. 2021+ Balko, Chodounský, Dobrinen, Hubička, Konečný, Vena, Zucker: universal partial order
6. 2022+ Balko, Chodounský, Dobrinen, Hubička, Konečný, Vena, Zucker: Free amalgamation classes in finite binary languages with finitely many forbidden substructures (96 pages draft)

Optimality

Known characterisations of big Ramsey degrees:

1. 1979 Devlin: the order of rationals
2. 2008 laflamme, Nguyen Van Th, Sauer: Ultrametric spaces
3. 2010 Laflamme, Sauer, Vuskanovic: the Rado graph
4. 2010 Balko, Chodounský, Hubička, Konečný, Vena, Zucker; independenty Dobrinen: Triangle free graphs
5. 2021+ Balko, Chodounský, Dobrinen, Hubička, Konečný, Vena, Zucker: universal partial order
6. 2022+ Balko, Chodounský, Dobrinen, Hubička, Konečný, Vena, Zucker: Free amalgamation classes in finite binary languages with finitely many forbidden substructures (96 pages draft)

So far the main application of such result is the completion flow introduced in:
A. Zucker, Big Ramsey degrees and topological dynamics, Groups Geom. Dyn., 2018.

Optimality

Known characterisations of big Ramsey degrees:

1. 1979 Devlin: the order of rationals
2. 2008 laflamme, Nguyen Van Th, Sauer: Ultrametric spaces
3. 2010 Laflamme, Sauer, Vuskanovic: the Rado graph
4. 2010 Balko, Chodounský, Hubička, Konečný, Vena, Zucker; independenty Dobrinen: Triangle free graphs
5. 2021+ Balko, Chodounský, Dobrinen, Hubička, Konečný, Vena, Zucker: universal partial order
6. 2022+ Balko, Chodounský, Dobrinen, Hubička, Konečný, Vena, Zucker: Free amalgamation classes in finite binary languages with finitely many forbidden substructures (96 pages draft)

So far the main application of such result is the completion flow introduced in:
A. Zucker, Big Ramsey degrees and topological dynamics, Groups Geom. Dyn., 2018.

We expect that understanding these types will help to develop structural Ellentuck-type theorems as well as help in understanding other aspects of homogeneous structures (such as the Cherlin-Lachlan classification programme).

Devlin-type

Given n, the big Ramsey degree of linear order of size n is to the number of Devlin-types.

Notation:

- $\Sigma^{<\omega}$ is the set of all finite words in alphabet Σ.
- Given $S \subseteq \Sigma^{<\omega}$ by $\overline{S}$ we denote the set of all initial segments of words in S.
- By $\overline{S}_i$ we denote the set of all initial segments of S of length i.
- By $w^\frown c$ we denote word w extended by character c (concatenation).
- $S^\frown c = \{w^\frown c : w \in S\}$.

Devlin-type

Given n, the big Ramsey degree of linear order of size n is to the number of Devlin-types.

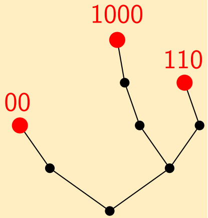
Notation:

- $\Sigma^{<\omega}$ is the set of all finite words in alphabet Σ.
- Given $S \subseteq \Sigma^{<\omega}$ by $\overline{S}$ we denote the set of all initial segments of words in S.
- By $\overline{S}_i$ we denote the set of all initial segments of S of length i.
- By $w^\frown c$ we denote word w extended by character c (concatenation).
- $S^\frown c = \{w^\frown c : w \in S\}$.

Definition (Devlin-type, alternative definition)

A Devlin-type is any subset S of $2^{<\omega}$ that is an antichain and for every $\ell \leq \max_{w\in S}|w|$ precisely one of the following happens:

1. **Leaf:** There is $w \in S_\ell$ such that $\overline{S}_{\ell+1} = (\overline{S}_\ell \setminus \{w\})^\frown 0$.
2. **Branching**: There is $w \in \overline{S}_\ell$ such that $\overline{S}_{\ell+1} = w^\frown 1 \cup (\overline{S}_\ell)^\frown 0$.

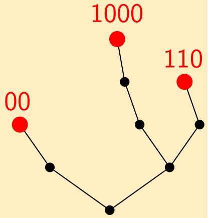


Type of a Rado graph

Definition (Devlin-type)

A Devlin-type is any subset S of $2^{<\omega}$ that is an antichain and for every $\ell \leq \max_{w \in S} |w|$ precisely one of the following happens:

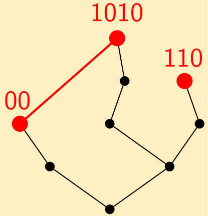
1. **Leaf:** There is $w \in S_\ell$ such that $\overline{S}_{\ell+1} = (\overline{S}_\ell \setminus \{w\})^\frown 0$.
2. **Branching**: There is $w \in \overline{S}_\ell$ such that $\overline{S}_{\ell+1} = w^\frown 1 \cup (\overline{S}_\ell)^\frown 0$.



Definition (Rado graph-type, Laflamme–Sauer–Vuksanovic)

A Rado graph-type is any subset S of $2^{<\omega}$ that is an antichain and for every $\ell \leq \max_{w \in S} |w|$ precisely one of the following happens:

1. **Leaf:** There is $w \in S_\ell$ such that $\overline{S}_{\ell+1}$ has precisely one successor of each of $\overline{S}_\ell \setminus \{w\}$.
2. **Branching**: There is $w \in \overline{S}_\ell$ such that $\overline{S}_{\ell+1} = w^\frown 1 \cup (\overline{S}_\ell)^\frown 0$.



Canonizing embeddings

Definition (Recall: graph G)

We will consider graph **G**:

1. Vertices: $2^{<\omega}$
2. Vertices $a, b \in 2^{<\omega}$ satisfying $|a| < |b|$ forms and edge if and only if $b(|a|) = 1$.
3. There are no other edges.

Proposition (On canonical forms of embeddings from G to G)

Let $f\colon \mathbf{G} \to \mathbf{G}$ be an (graph and not necessarily tree) embedding. Then there exists a strong subtree S of T and a sequence $(N_i)_{i\in\omega}$ of integers satisfying:

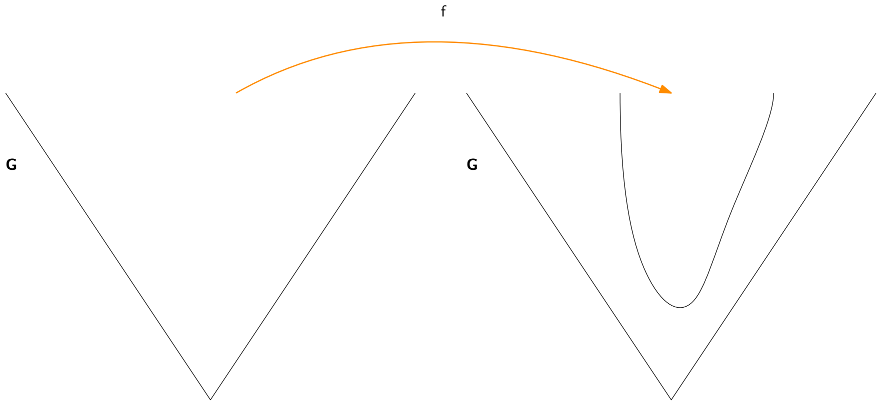
1. for every $a \in S$ it holds that $N_{|a|} \leq |f(\varphi_S(a))| < N_{|a|+1}$,
2. for every $a, b \in S$ and every $\ell < \min(|a|, |b|)$ such that $a|_\ell = b|_\ell$ it holds that $f(\varphi_S(a))|_{N_\ell} = f(\varphi_S(b))|_{N_\ell}$.

Canonizing embeddings

Proposition (On canonical forms of embeddings from $\mathbf{G}$ to $\mathbf{G}$)

Let $f\colon \mathbf{G} \to \mathbf{G}$ be an (graph and not necessarily tree) embedding. Then there exists a strong subtree S of T and a sequence $(N_i)_{i\in\omega}$ of integers satisfying:

1. for every $a \in S$ it holds that $N_{|a|} \leq |f(\varphi_S(a))| < N_{|a|+1}$,
2. for every $a, b \in S$ and every $\ell < \min(|a|, |b|)$ such that $a|_\ell = b|_\ell$ it holds that $f(\varphi_S(a))|_{N_\ell} = f(\varphi_S(b))|_{N_\ell}$.

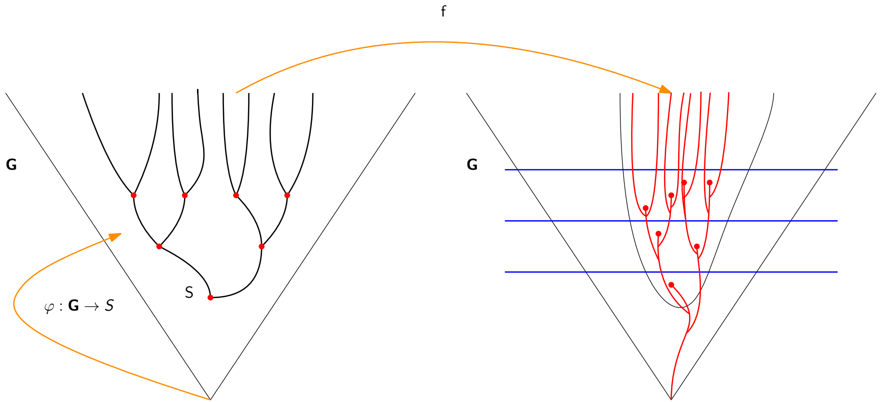


Canonizing embeddings

Proposition (On canonical forms of embeddings from **G** to **G**)

Let $f\colon \mathbf{G} \to \mathbf{G}$ be an (graph and not necessarily tree) embedding. Then there exists a strong subtree S of T and a sequence $(N_i)_{i\in\omega}$ of integers satisfying:

1. for every $a \in S$ it holds that $N_{|a|} \leq |f(\varphi_S(a))| < N_{|a|+1}$,
2. for every $a, b \in S$ and every $\ell < \min(|a|, |b|)$ such that $a|_\ell = b|_\ell$ it holds that $f(\varphi_S(a))|_{N_\ell} = f(\varphi_S(b))|_{N_\ell}$.

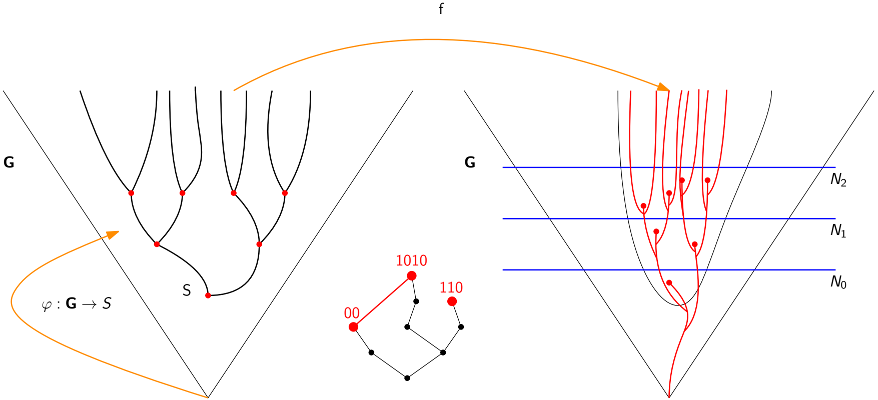


Canonizing embeddings

Proposition (On canonical forms of embeddings from **G** to **G**)

Let $f\colon \mathbf{G} \to \mathbf{G}$ be an (graph and not necessarily tree) embedding. Then there exists a strong subtree S of T and a sequence $(N_i)_{i\in\omega}$ of integers satisfying:

1. for every $a \in S$ it holds that $N_{|a|} \leq |f(\varphi_S(a))| < N_{|a|+1}$,
2. for every $a, b \in S$ and every $\ell < \min(|a|, |b|)$ such that $a|_\ell = b|_\ell$ it holds that $f(\varphi_S(a))|_{N_\ell} = f(\varphi_S(b))|_{N_\ell}$.

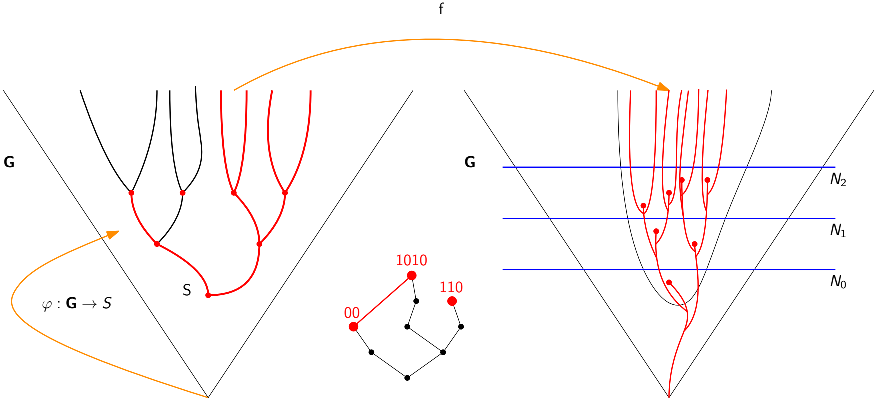


Canonizing embeddings

Proposition (On canonical forms of embeddings from **G** to **G**)

Let $f\colon \mathbf{G} \to \mathbf{G}$ be an (graph and not necessarily tree) embedding. Then there exists a strong subtree S of T and a sequence $(N_i)_{i\in\omega}$ of integers satisfying:

1. for every $a \in S$ it holds that $N_{|a|} \leq |f(\varphi_S(a))| < N_{|a|+1}$,
2. for every $a, b \in S$ and every $\ell < \min(|a|, |b|)$ such that $a|_\ell = b|_\ell$ it holds that $f(\varphi_S(a))|_{N_\ell} = f(\varphi_S(b))|_{N_\ell}$.

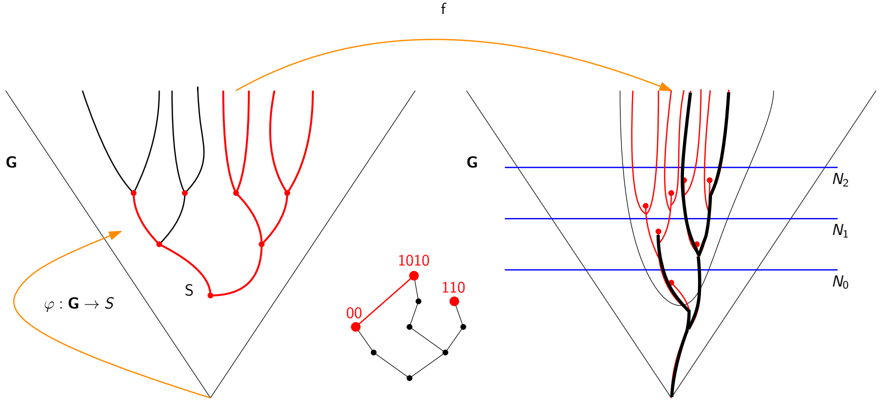


Canonizing embeddings

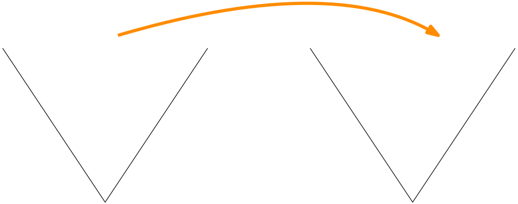
Proposition (On canonical forms of embeddings from $\mathbf{G}$ to $\mathbf{G}$)

Let $f\colon \mathbf{G} \to \mathbf{G}$ be an (graph and not necessarily tree) embedding. Then there exists a strong subtree S of T and a sequence $(N_i)_{i\in\omega}$ of integers satisfying:

1. for every $a \in S$ it holds that $N_{|a|} \leq |f(\varphi_S(a))| < N_{|a|+1}$,
2. for every $a, b \in S$ and every $\ell < \min(|a|, |b|)$ such that $a|_\ell = b|_\ell$ it holds that $f(\varphi_S(a))|_{N_\ell} = f(\varphi_S(b))|_{N_\ell}$.

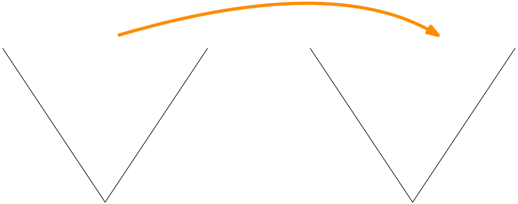


Proof



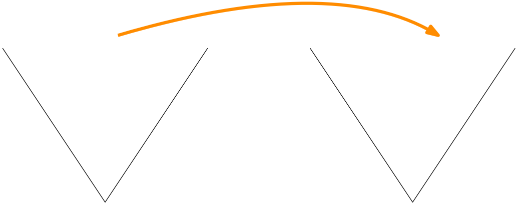
1. Fix embedding $f\colon \mathbf{G} \to \mathbf{G}$. Produce a sequences of sub-trees $(S_i)_{i\in\omega}$ and integers $(N_i)_{i\in\omega}$.
2. Put $S_0 = T$, $N_0 = 0$.

Proof



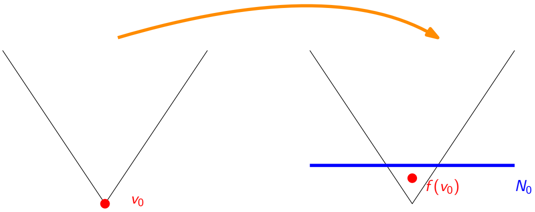
1. Fix embedding $f\colon \mathbf{G} \to \mathbf{G}$. Produce a sequences of sub-trees $(S_i)_{i\in\omega}$ and integers $(N_i)_{i\in\omega}$.
2. Put $S_0 = T$, $N_0 = 0$.
3. Now assume that S_i and N_i are already constructed. Put: $N_{i+1} = \max\{|f(\varphi_{S_i}(a))| + 1 : \text{a is in level } i \text{ of } S_i\}$.
4. Let $\mathcal{T}_i$ be the collection of all strong sub-trees of S_i of depth i such that first $i-1$ levels are precisely first $i-1$ levels of S_i.

Proof



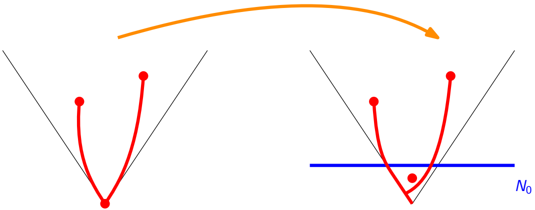
1. Fix embedding $f\colon \mathbf{G} \to \mathbf{G}$. Produce a sequences of sub-trees $(S_i)_{i\in\omega}$ and integers $(N_i)_{i\in\omega}$.
2. Put $S_0 = T$, $N_0 = 0$.
3. Now assume that S_i and N_i are already constructed. Put: $N_{i+1} = \max\{|f(\varphi_{S_i}(a))| + 1 : \text{a is in level } i \text{ of } S_i\}$.
4. Let $\mathcal{T}_i$ be the collection of all strong sub-trees of S_i of depth i such that first $i-1$ levels are precisely first $i-1$ levels of S_i.
5. Define colouring χ of $\mathcal{T}_i$: Given $T' \in \mathcal{T}^i$ let $(a_0, a_1, \dots, a_{n-1})$ be an enumeration of all leafs (lexicographically). Now let $\chi(T')$ be the function from $\{0, 1, \dots, n-1\}$ defined by $\chi(T')(j) = f(\varphi_{S_i}(a_j))|_{N_{i+1}}$.
6. Apply (product) Millken theorem to obtain S_{i+1}.

Proof



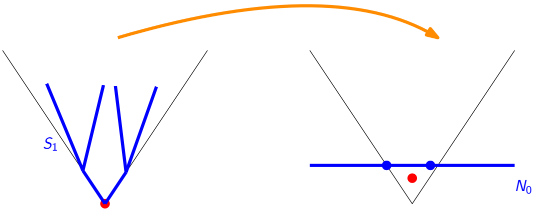
1. Fix embedding $f\colon \mathbf{G} \to \mathbf{G}$. Produce a sequences of sub-trees $(S_i)_{i\in\omega}$ and integers $(N_i)_{i\in\omega}$.
2. Put $S_0 = T$, $N_0 = 0$.
3. Now assume that S_i and N_i are already constructed. Put: $N_{i+1} = \max\{|f(\varphi_{S_i}(a))| + 1 : \text{a is in level } i \text{ of } S_i\}$.
4. Let $\mathcal{T}_i$ be the collection of all strong sub-trees of S_i of depth i such that first $i-1$ levels are precisely first $i-1$ levels of S_i.
5. Define colouring χ of $\mathcal{T}_i$: Given $T' \in \mathcal{T}^i$ let $(a_0, a_1, \ldots, a_{n-1})$ be an enumeration of all leafs (lexicographically). Now let $\chi(T')$ be the function from $\{0, 1, \ldots, n-1\}$ defined by $\chi(T')(j) = f(\varphi_{S_i}(a_j))|_{N_{i+1}}$.
6. Apply (product) Millken theorem to obtain S_{i+1}.

Proof



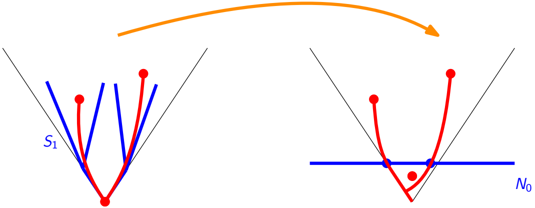
1. Fix embedding $f\colon \mathbf{G} \to \mathbf{G}$. Produce a sequences of sub-trees $(S_i)_{i\in\omega}$ and integers $(N_i)_{i\in\omega}$.
2. Put $S_0 = T$, $N_0 = 0$.
3. Now assume that S_i and N_i are already constructed. Put: $N_{i+1} = \max\{|f(\varphi_{S_i}(a))| + 1 : \text{a is in level } i \text{ of } S_i\}$.
4. Let $\mathcal{T}_i$ be the collection of all strong sub-trees of S_i of depth i such that first $i-1$ levels are precisely first $i-1$ levels of S_i.
5. Define colouring χ of $\mathcal{T}_i$: Given $T' \in \mathcal{T}^i$ let $(a_0, a_1, \ldots, a_{n-1})$ be an enumeration of all leafs (lexicographically). Now let $\chi(T')$ be the function from $\{0, 1, \ldots, n-1\}$ defined by $\chi(T')(j) = f(\varphi_{S_i}(a_j))|_{N_{i+1}}$.
6. Apply (product) Millken theorem to obtain S_{i+1}.

Proof



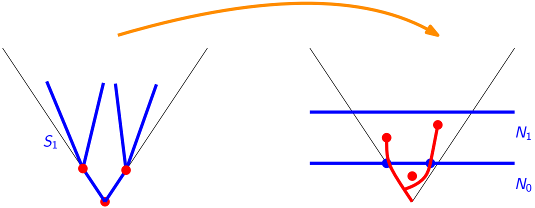
1. Fix embedding $f\colon \mathbf{G} \to \mathbf{G}$. Produce a sequences of sub-trees $(S_i)_{i\in\omega}$ and integers $(N_i)_{i\in\omega}$.
2. Put $S_0 = T$, $N_0 = 0$.
3. Now assume that S_i and N_i are already constructed. Put: $N_{i+1} = \max\{|f(\varphi_{S_i}(a))| + 1 : a \text{ is in level } i \text{ of } S_i\}$.
4. Let $\mathcal{T}_i$ be the collection of all strong sub-trees of S_i of depth i such that first $i-1$ levels are precisely first $i-1$ levels of S_i.
5. Define colouring χ of $\mathcal{T}_i$: Given $T' \in \mathcal{T}^i$ let $(a_0, a_1, \ldots, a_{n-1})$ be an enumeration of all leafs (lexicographically). Now let $\chi(T')$ be the function from $\{0, 1, \ldots, n-1\}$ defined by $\chi(T')(j) = f(\varphi_{S_i}(a_j))|_{N_{i+1}}$.
6. Apply (product) Millken theorem to obtain S_{i+1}.

Proof



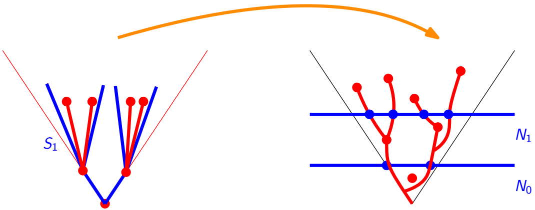
1. Fix embedding $f\colon \mathbf{G} \to \mathbf{G}$. Produce a sequences of sub-trees $(S_i)_{i\in\omega}$ and integers $(N_i)_{i\in\omega}$.
2. Put $S_0 = T$, $N_0 = 0$.
3. Now assume that S_i and N_i are already constructed. Put: $N_{i+1} = \max\{|f(\varphi_{S_i}(a))| + 1 : a \text{ is in level } i \text{ of } S_i\}$.
4. Let $\mathcal{T}_i$ be the collection of all strong sub-trees of S_i of depth i such that first $i-1$ levels are precisely first $i-1$ levels of S_i.
5. Define colouring χ of $\mathcal{T}_i$: Given $T' \in \mathcal{T}^i$ let $(a_0, a_1, \ldots, a_{n-1})$ be an enumeration of all leafs (lexicographically). Now let $\chi(T')$ be the function from $\{0, 1, \ldots, n-1\}$ defined by $\chi(T')(j) = f(\varphi_{S_i}(a_j))|_{N_{i+1}}$.
6. Apply (product) Millken theorem to obtain S_{i+1}.

Proof



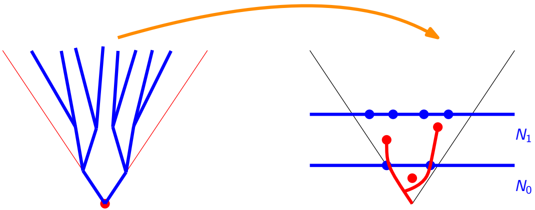
1. Fix embedding $f\colon \mathbf{G} \to \mathbf{G}$. Produce a sequences of sub-trees $(S_i)_{i\in\omega}$ and integers $(N_i)_{i\in\omega}$.
2. Put $S_0 = T$, $N_0 = 0$.
3. Now assume that S_i and N_i are already constructed. Put: $N_{i+1} = \max\{|f(\varphi_{S_i}(a))| + 1 : a \text{ is in level } i \text{ of } S_i\}$.
4. Let $\mathcal{T}_i$ be the collection of all strong sub-trees of S_i of depth i such that first $i-1$ levels are precisely first $i-1$ levels of S_i.
5. Define colouring χ of $\mathcal{T}_i$: Given $T' \in \mathcal{T}^i$ let $(a_0, a_1, \ldots, a_{n-1})$ be an enumeration of all leafs (lexicographically). Now let $\chi(T')$ be the function from $\{0, 1, \ldots, n-1\}$ defined by $\chi(T')(j) = f(\varphi_{S_i}(a_j))|_{N_{i+1}}$.
6. Apply (product) Millken theorem to obtain S_{i+1}.

Proof



1. Fix embedding $f\colon \mathbf{G} \to \mathbf{G}$. Produce a sequences of sub-trees $(S_i)_{i\in\omega}$ and integers $(N_i)_{i\in\omega}$.
2. Put $S_0 = T$, $N_0 = 0$.
3. Now assume that S_i and N_i are already constructed. Put: $N_{i+1} = \max\{|f(\varphi_{S_i}(a))| + 1 : a \text{ is in level } i \text{ of } S_i\}$.
4. Let $\mathcal{T}_i$ be the collection of all strong sub-trees of S_i of depth i such that first $i-1$ levels are precisely first $i-1$ levels of S_i.
5. Define colouring χ of $\mathcal{T}_i$: Given $T' \in \mathcal{T}^i$ let $(a_0, a_1, \ldots, a_{n-1})$ be an enumeration of all leafs (lexicographically). Now let $\chi(T')$ be the function from $\{0, 1, \ldots, n-1\}$ defined by $\chi(T')(j) = f(\varphi_{S_i}(a_j))|_{N_{i+1}}$.
6. Apply (product) Millken theorem to obtain S_{i+1}.

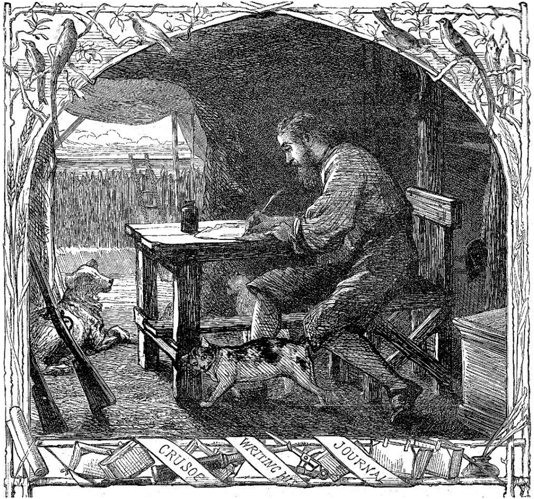
Proof



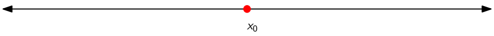
1. Fix embedding $f\colon \mathbf{G} \to \mathbf{G}$. Produce a sequences of sub-trees $(S_i)_{i\in\omega}$ and integers $(N_i)_{i\in\omega}$.
2. Put $S_0 = T$, $N_0 = 0$.
3. Now assume that S_i and N_i are already constructed. Put: $N_{i+1} = \max\{|f(\varphi_{S_i}(a))| + 1 : a \text{ is in level } i \text{ of } S_i\}$.
4. Let $\mathcal{T}_i$ be the collection of all strong sub-trees of S_i of depth i such that first $i-1$ levels are precisely first $i-1$ levels of S_i.
5. Define colouring χ of $\mathcal{T}_i$: Given $T' \in \mathcal{T}^i$ let $(a_0, a_1, \dots, a_{n-1})$ be an enumeration of all leafs (lexicographically). Now let $\chi(T')$ be the function from $\{0, 1, \dots, n-1\}$ defined by $\chi(T')(j) = f(\varphi_{S_i}(a_j))|_{N_{i+1}}$.
6. Apply (product) Millken theorem to obtain S_{i+1}.

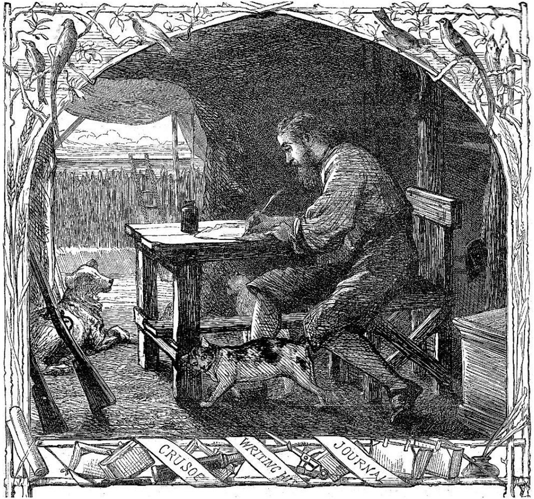

NADĚJE VRTAČEK

What are the types



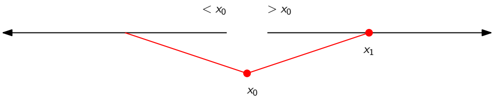
What are the types

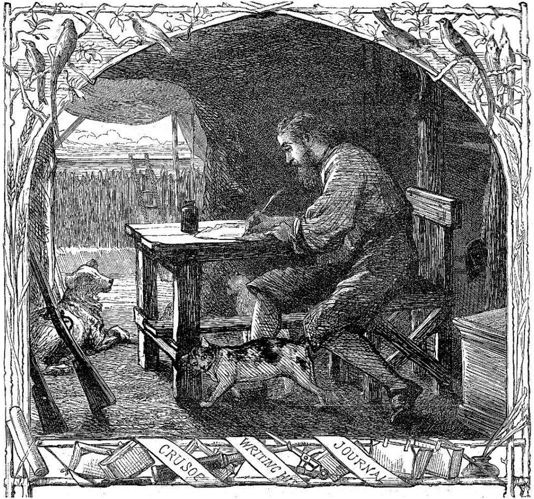




$$\mathscr{Vertex}\,(t_0 \to x_0),\ \mathscr{Branch}\,(t_0 \to t_1, t_2),$$

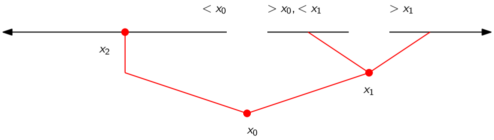
What are the types

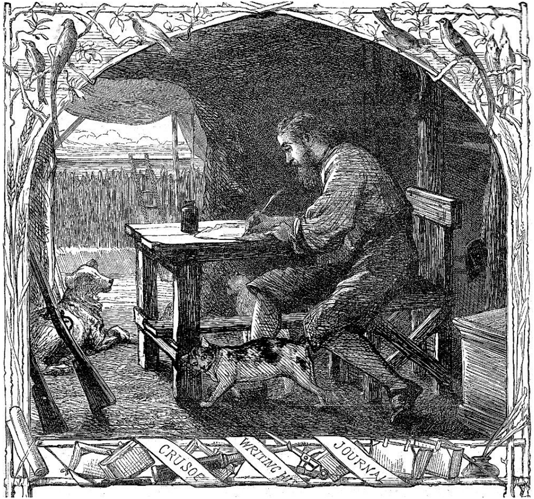




$$\mathscr{Vertex}\,(t_0 \to x_0),\ \mathscr{Branch}\,(t_0 \to t_1, t_2),$$
$$\mathscr{Vertex}\,(t_2 \to x_1),\ \mathscr{Branch}\,(t_2 \to t_3, t_4),$$

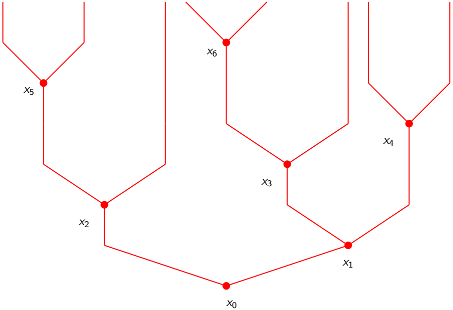
What are the types

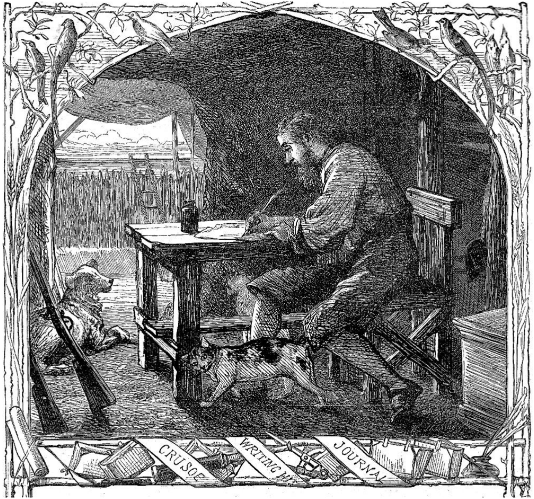




$$\mathscr{Vertex}\,(t_0 \to x_0),\ \mathscr{Branch}\,(t_0 \to t_1, t_2),$$
$$\mathscr{Vertex}\,(t_2 \to x_1),\ \mathscr{Branch}\,(t_2 \to t_3, t_4),$$
$$\mathscr{Vertex}\,(t_1 \to x_2),\ \mathscr{Branch}\,(t_1 \to t_5, t_6),$$

What are the types





$$\mathscr{Vertex}\,(t_0 \to x_0),\ \mathscr{Branch}\,(t_0 \to t_1, t_2),$$
$$\mathscr{Vertex}\,(t_2 \to x_1),\ \mathscr{Branch}\,(t_2 \to t_3, t_4),$$
$$\mathscr{Vertex}\,(t_1 \to x_2),\ \mathscr{Branch}\,(t_1 \to t_5, t_6),$$
$$\mathscr{Vertex}\,(t_3 \to x_3),\ \mathscr{Branch}\,(t_3 \to t_7, t_8),$$
$$\mathscr{Vertex}\,(t_4 \to x_4),\ \mathscr{Branch}\,(t_4 \to t_9, t_{10}),$$

Level-structures

Definition

Given level ℓ of the tree of types, we can consider its level-structure:

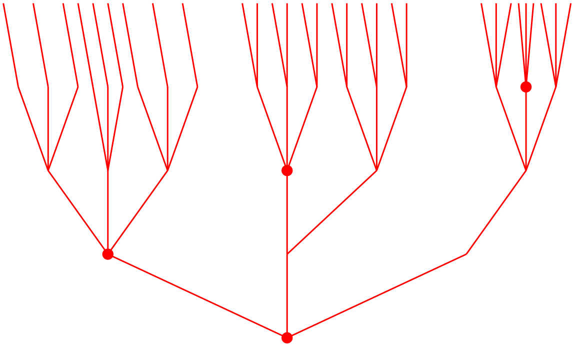
1. Vertices are nodes (types) of level ℓ.
2. We write $a \preceq b$ if it is true that $a' \leq b'$ for every successor a' of a and b' of b.
3. We write $a \trianglelefteq b$ if it is true that $a' \leq b'$ for some successor a' of a and b' of b.
4. We write $a \perp b$ if it is true that $a' \perp b'$ for every successor a' of a and b' of b.

Level-structures

Definition

Given level ℓ of the tree of types, we can consider its level-structure:

1. Vertices are nodes (types) of level ℓ.
2. We write $a \preceq b$ if it is true that $a' \leq b'$ for every successor a' of a and b' of b.
3. We write $a \trianglelefteq b$ if it is true that $a' \leq b'$ for some successor a' of a and b' of b.
4. We write $a \perp b$ if it is true that $a' \perp b'$ for every successor a' of a and b' of b.

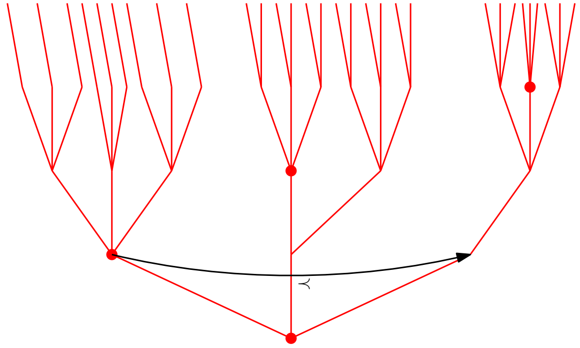


Level-structures

Definition

Given level ℓ of the tree of types, we can consider its level-structure:

1. Vertices are nodes (types) of level ℓ.
2. We write $a \preceq b$ if it is true that $a' \leq b'$ for every successor a' of a and b' of b.
3. We write $a \trianglelefteq b$ if it is true that $a' \leq b'$ for some successor a' of a and b' of b.
4. We write $a \perp b$ if it is true that $a' \perp b'$ for every successor a' of a and b' of b.

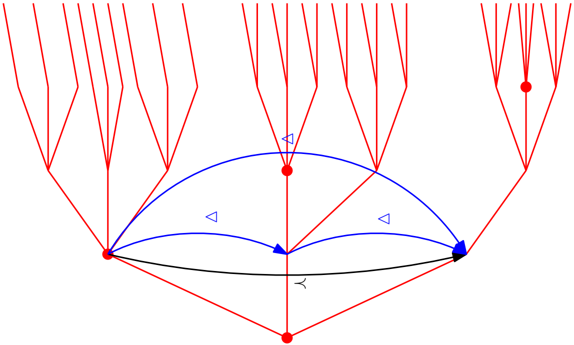


Level-structures

Definition

Given level ℓ of the tree of types, we can consider its level-structure:

1. Vertices are nodes (types) of level ℓ.
2. We write $a \preceq b$ if it is true that $a' \leq b'$ for every successor a' of a and b' of b.
3. We write $a \trianglelefteq b$ if it is true that $a' \leq b'$ for some successor a' of a and b' of b.
4. We write $a \perp b$ if it is true that $a' \perp b'$ for every successor a' of a and b' of b.

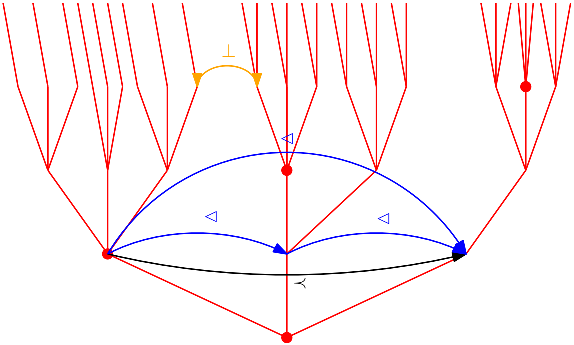


Level-structures

Definition

Given level ℓ of the tree of types, we can consider its level-structure:

1. Vertices are nodes (types) of level ℓ.
2. We write $a \preceq b$ if it is true that $a' \leq b'$ for every successor a' of a and b' of b.
3. We write $a \trianglelefteq b$ if it is true that $a' \leq b'$ for some successor a' of a and b' of b.
4. We write $a \perp b$ if it is true that $a' \perp b'$ for every successor a' of a and b' of b.



Level-structures

Definition

Given level ℓ of the tree of types, we can consider its level-structure:

1. Vertices are nodes (types) of level ℓ.
2. We write $a \preceq b$ if it is true that $a' \leq b'$ for every successor a' of a and b' of b.
3. We write $a \trianglelefteq b$ if it is true that $a' \leq b'$ for some successor a' of a and b' of b.
4. We write $a \perp b$ if it is true that $a' \perp b'$ for every successor a' of a and b' of b.

Fun fact

It turns out that both $\preceq$ and $\trianglelefteq$ are partial orders and whenever $a \preceq b$ also $a \trianglelefteq b$. One can think of the level structure $(A, \preceq, \trianglelefteq)$ as of an finite approximation of the infinite partial order located above the given level.

Definition (Poset-type)

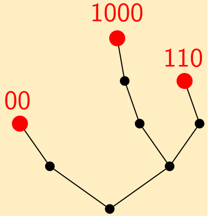
A set $S \subseteq \{\mathrm{L}, \mathrm{X}, \mathrm{R}\}^*$ is called a poset-type if precisely one of the following four conditions is satisfied for every level ℓ with $0 \leq \ell < \max_{w \in S} |w|$:

1. **Leaf:** There is $w \in \overline{S}_\ell$ related to every $u \in \overline{S}_\ell \setminus \{w\}$ and $\overline{S}_{\ell+1} = (\overline{S}_\ell \setminus \{w\})^\frown \mathrm{X}$.
2. **Branching:** There is $w \in \overline{S}_\ell$ such that
$$\overline{S}_{\ell+1} = \{z \in \overline{S}_\ell : z <_{\text{lex}} w\}^\frown \mathrm{X} \cup \{w^\frown \mathrm{X}, w^\frown \mathrm{R}\} \cup \{z \in \overline{S}_\ell : w <_{\text{lex}} z\}^\frown \mathrm{R}.$$
3. **New $\perp$:** There are unrelated words $v <_{\text{lex}} w \in \overline{S}_\ell$ such that
$$\begin{aligned}\overline{S}_{\ell+1} = {} & \{z \in \overline{S}_\ell : z <_{\text{lex}} v\}^\frown \mathrm{X} \cup \{v^\frown \mathrm{R}\} \cup \{z \in \overline{S}_\ell : v <_{\text{lex}} z <_{\text{lex}} w \text{ and } z \perp v\}^\frown \mathrm{X} \\ & \cup \{z \in \overline{S}_\ell : v <_{\text{lex}} z <_{\text{lex}} w \text{ and } z \not\perp v\}^\frown \mathrm{R} \cup \{w^\frown \mathrm{X}\} \cup \{z \in \overline{S}_\ell : w <_{\text{lex}} z\}^\frown \mathrm{R}.\end{aligned}$$
Moreover for every $u \in \overline{S}_\ell$, $v <_{\text{lex}} u <_{\text{lex}} w$ implies that at least one of $u \perp v$ or $u \perp w$ holds.
4. **New $\prec$:** There are unrelated words $v <_{\text{lex}} w \in \overline{S}_\ell$ such that
$$\begin{aligned}\overline{S}_{\ell+1} = {} & \{z \in \overline{S}_\ell : z <_{\text{lex}} v \text{ and } z \perp v\}^\frown \mathrm{X} \cup \{z \in \overline{S}_\ell : z <_{\text{lex}} v \text{ and } z \not\perp v\}^\frown \mathrm{L} \cup \{v^\frown \mathrm{L}\} \\ & \cup \{z \in \overline{S}_\ell : v <_{\text{lex}} z <_{\text{lex}} w\}^\frown \mathrm{X} \cup \{w^\frown \mathrm{R}\} \cup \{z \in \overline{S}_\ell : w <_{\text{lex}} z \text{ and } w \perp z\}^\frown \mathrm{X} \\ & \cup \{z \in \overline{S}_\ell : w <_{\text{lex}} z \text{ and } w \not\perp z\}^\frown \mathrm{R}.\end{aligned}$$
Moreover for every $u \in \overline{S}_\ell$ such that $u <_{\text{lex}} v$, at least one of $u \preceq w$ or $u \perp v$ holds.
Symmetrically for every $u \in \overline{S}_\ell$ such that $w <_{\text{lex}} u$, at least one of $v \preceq u$ or $w \perp u$ holds.

Definition (Devlin-type)

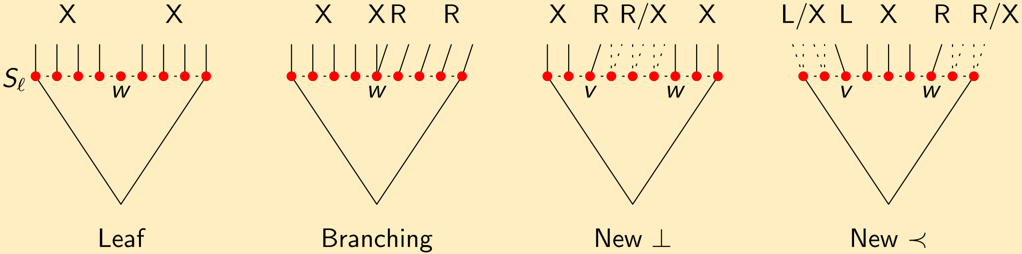
A Devlin-type is any subset S of $\{0,1\}^*$ such that for every $\ell \leq \max_{w \in S} |w|$ precisely one of the following happens:

1. **Leaf:** There is $w \in \overline{S}_\ell$ such that $\overline{S}_{\ell+1} = (\overline{S}_\ell \setminus \{w\})^\frown 0$.
2. **Branching**: There is $w \in \overline{S}_\ell$ such that $\overline{S}_{\ell+1} = w^\frown 1 \cup (\overline{S}_\ell)^\frown 0$.



Definition (Poset-type)

A Poset-type is any subset S of $\{\mathrm{L}, \mathrm{X}, \mathrm{R}\}^*$ such that for every $\ell \leq \max_{w \in S} |w|$ precisely one of the following happens:



Main result

Given a finite partial order $(A, \leq)$, we let $T(A, \leq)$ be the set of all poset-types S such that $(S, \preceq)$ is isomorphic to $(A, \leq)$.

Theorem (M. Balko, D. Chodounský, N. Dobrinen, J. H., M. Konečný, L. Vena, A. Zucker)

For every finite partial order $(O, \leq)$, the big Ramsey degree of $(O, \leq)$ in the universal partial order $(P, \leq)$ equals $|T(O, \leq)| \cdot |\mathrm{Aut}(O, \leq)|$.

Main result

Given a finite partial order $(A, \leq)$, we let $T(A, \leq)$ be the set of all poset-types S such that $(S, \preceq)$ is isomorphic to $(A, \leq)$.

Theorem (M. Balko, D. Chodounský, N. Dobrinen, J. H., M. Konečný, L. Vena, A. Zucker)

For every finite partial order $(O, \leq)$, the big Ramsey degree of $(O, \leq)$ in the universal partial order $(P, \leq)$ equals $|T(O, \leq)| \cdot |\mathrm{Aut}(O, \leq)|$.

Example

Denote by $\mathbf{A}_n$ the anti-chain with with n vertices and by $\mathbf{C}_n$ the chain with n vertices.

$$T(\mathbf{A}_1) = T(\mathbf{C}_1) = \{\emptyset\}$$

$$T(\mathbf{A}_2) = \{\{\mathrm{XR}, \mathrm{RXX}\}, \{\mathrm{XRX}, \mathrm{RX}\}\}$$
$$T(\mathbf{C}_2) = \{\{\mathrm{XL}, \mathrm{RRX}\}, \{\mathrm{XLX}, \mathrm{RR}\}\}$$

Main result

Given a finite partial order $(A, \leq)$, we let $T(A, \leq)$ be the set of all poset-types S such that $(S, \preceq)$ is isomorphic to $(A, \leq)$.

Theorem (M. Balko, D. Chodounský, N. Dobrinen, J. H., M. Konečný, L. Vena, A. Zucker)

For every finite partial order $(O, \leq)$, the big Ramsey degree of $(O, \leq)$ in the universal partial order $(P, \leq)$ equals $|T(O, \leq)| \cdot |\mathrm{Aut}(O, \leq)|$.

Example

Denote by $\mathbf{A}_n$ the anti-chain with with n vertices and by $\mathbf{C}_n$ the chain with n vertices.

$$T(\mathbf{A}_1) = T(\mathbf{C}_1) = \{\emptyset\}$$

$$T(\mathbf{A}_2) = \{\{\mathrm{XR}, \mathrm{RXX}\}, \{\mathrm{XRX}, \mathrm{RX}\}\}$$
$$T(\mathbf{C}_2) = \{\{\mathrm{XL}, \mathrm{RRX}\}, \{\mathrm{XLX}, \mathrm{RR}\}\}$$

$$|T(\mathbf{C}_3)| = 52, |T(\mathbf{C}_4)| = 11000,$$

$$|T(\mathbf{A}_3)| = 84, |T(\mathbf{A}_4)| = 75642$$

Main result

Given a finite partial order $(A, \leq)$, we let $T(A, \leq)$ be the set of all poset-types S such that $(S, \preceq)$ is isomorphic to $(A, \leq)$.

Theorem (M. Balko, D. Chodounský, N. Dobrinen, J. H., M. Konečný, L. Vena, A. Zucker)

For every finite partial order $(O, \leq)$, the big Ramsey degree of $(O, \leq)$ in the universal partial order $(P, \leq)$ equals $|T(O, \leq)| \cdot |\mathrm{Aut}(O, \leq)|$.

Example

Denote by $\mathbf{A}_n$ the anti-chain with with n vertices and by $\mathbf{C}_n$ the chain with n vertices.

$$T(\mathbf{A}_1) = T(\mathbf{C}_1) = \{\emptyset\}$$

$$T(\mathbf{A}_2) = \{\{\mathrm{XR}, \mathrm{RXX}\}, \{\mathrm{XRX}, \mathrm{RX}\}\}$$
$$T(\mathbf{C}_2) = \{\{\mathrm{XL}, \mathrm{RRX}\}, \{\mathrm{XLX}, \mathrm{RR}\}\}$$

$$|T(\mathbf{C}_3)| = 52, |T(\mathbf{C}_4)| = 11000,$$

$$|T(\mathbf{A}_3)| = 84, |T(\mathbf{A}_4)| = 75642$$

Overall there are:

- 1 poset-types a vertex,
- 4 poset-types of posets of size 2,
- 464 poset-types of posets of size 3,
- 1874880 poset-types of posets of size 4.

Approximating posets

Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting "finished vertices".
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

Approximating posets

Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting "finished vertices".
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

Definition

Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

Approximating posets

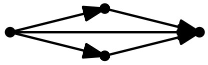
Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting “finished vertices”.
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Goal



Approximation

Definition

Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

Approximating posets

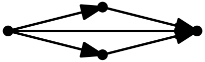
$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting "finished vertices".
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \not\trianglelefteq b$, $b \not\trianglelefteq a$.)

Goal



Approximation

Definition

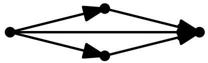
Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

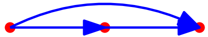
Approximating posets

$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Goal



Approximation



Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting “finished vertices”.
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \not\trianglelefteq b$, $b \not\trianglelefteq a$.)

Definition

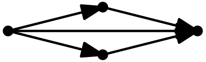
Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

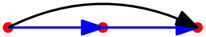
Approximating posets

$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Goal



Approximation



Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting "finished vertices".
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \not\trianglelefteq b$, $b \not\trianglelefteq a$.)

Definition

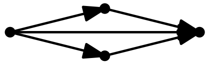
Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

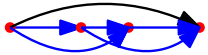
Approximating posets

$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Goal



Approximation



Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting “finished vertices”.
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

Definition

Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

Approximating posets

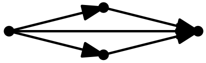
$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Definition (Approximate poset)

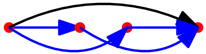
An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting “finished vertices”.
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

Goal



Approximation



Definition

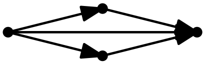
Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

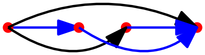
Approximating posets

$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Goal



Approximation



Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting "finished vertices".
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

Definition

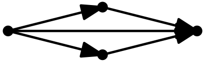
Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

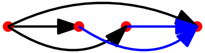
Approximating posets

$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Goal



Approximation



Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting “finished vertices”.
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

Definition

Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

Approximating posets

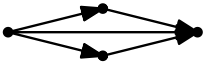
$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Definition (Approximate poset)

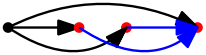
An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting “finished vertices”.
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

Goal



Approximation



Definition

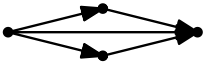
Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

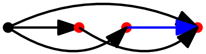
Approximating posets

$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Goal



Approximation



Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting “finished vertices”.
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \not\trianglelefteq b$, $b \not\trianglelefteq a$.)

Definition

Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

Approximating posets

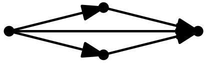
$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Definition (Approximate poset)

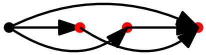
An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting “finished vertices”.
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

Goal



Approximation



Definition

Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

Approximating posets

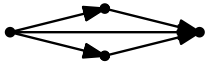
$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Definition (Approximate poset)

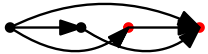
An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting "finished vertices".
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

Goal



Approximation



Definition

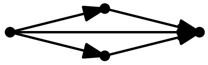
Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

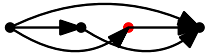
Approximating posets

$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Goal



Approximation



Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting "finished vertices".
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \not\trianglelefteq b$, $b \not\trianglelefteq a$.)

Definition

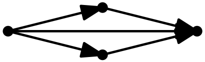
Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

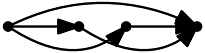
Approximating posets

$\preceq$ is black, $\trianglelefteq$ is blue
P is black, $\neg P$ is red

Goal



Approximation



Definition (Approximate poset)

An approximate poset is a structure $(A, \preceq, \trianglelefteq, P)$ where

1. both $\preceq$ and $\trianglelefteq$ are partial order.
2. P is an unary predicate denoting “finished vertices”.
3. $\preceq \subseteq \trianglelefteq$ (whenever $a \preceq b$ also $a \trianglelefteq b$).
4. $a \preceq b \trianglelefteq c \implies a \preceq c$ and $a \trianglelefteq b \preceq c \implies a \preceq c$.
5. If $a \in p$ then for every $b \in A$, $b \neq a$ it holds one of $a \preceq b$, $b \preceq a$ or $a \perp b$ ($a \perp b$ is a shortcut for $a \ntrianglelefteq b$, $b \ntrianglelefteq a$.)

Definition

Poset-type of a poset $(B, \leq)$ is equivalently sequence of approximations of $(B, \leq)$ where on each step one of the following happens:

1. **Leaf**: New vertex is added to predicate P.
2. **Branching**: A vertex u is split into vertices $u_1 \trianglelefteq u_2$.
3. **New** $\perp$: A pair is removed from relation $\trianglelefteq$.
4. **New** $\prec$: A new pair is added to relation $\preceq$.

Definition (Trianglefree-type)

A set $S \subseteq \Sigma^*$ is called a *triangle-free-type* if $S = \overline{S}$ and precisely one of the following four conditions is satisfied for every i with $0 \leq i < \max_{w \in S} |w|$:

1. **Leaf:** There is $w \in S_i$ containing at least one 1 such that
$$S_{i+1} = \{z \in S_i \setminus \{w\} : z \perp w\}^\frown 0 \cup \{z \in S_i \setminus \{w\} : z \not\perp w\}^\frown 1.$$
2. **Branching:** There is $w \in S_i$ such that
$$S_{i+1} = S_i^\frown 0 \cup \{w\}^\frown 1.$$
3. **First neighbour:** There is $w \in S_i$ containing no 1 such that
$$S_{i+1} = (S_i \setminus \{w\})^\frown 0 \cup \{w\}^\frown 1.$$
4. **New $\perp$:** There are distinct words $v, w \in S_i$ each containing at least one 1 satisfying $v \not\perp w$ such that
$$S_{i+1} = (S_i \setminus \{v, w\})^\frown 0 \cup \{v, w\}^\frown 1.$$

Thank you!

Sleeping Child, Fred Payne Clatworthy, Autochrome, 7 x 5 inches, c1916
Mark Jacobs Collection