Ramsey theorem for trees with a successor operation

Jan Hubička

Department of Applied Mathematics
Charles University
Prague

Joint work with Martin Balko, Natasha Dobrinen, David Chodounský, Matěj Konečný, Jaroslav Nešetřil, Lluis Vena, Andy Zucker

Toposym 2022, Prague

Known big Ramsey results by proof techniques

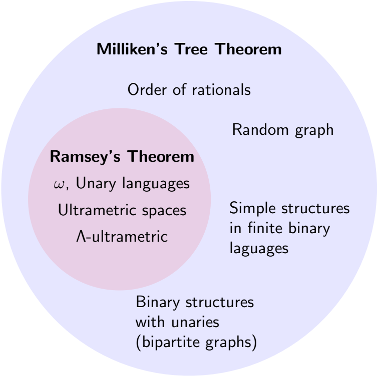
Ramsey's Theorem

ω, Unary languages

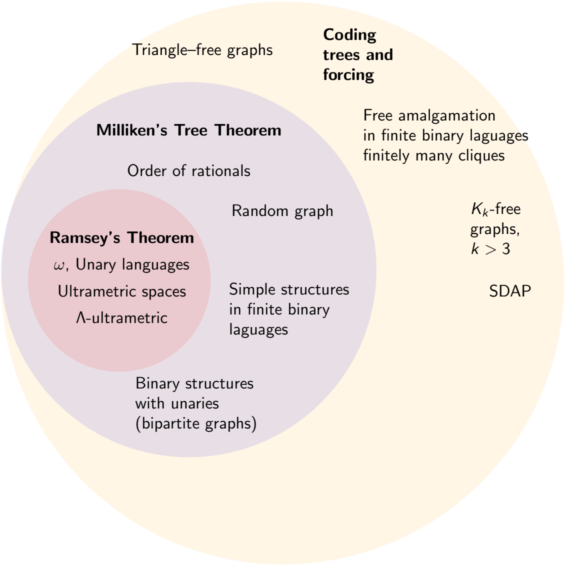
Ultrametric spaces

Λ-ultrametric

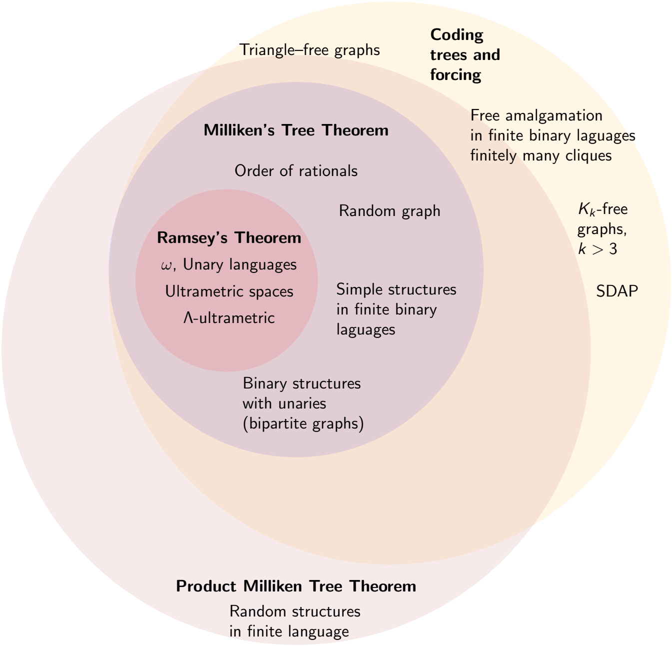
Known big Ramsey results by proof techniques



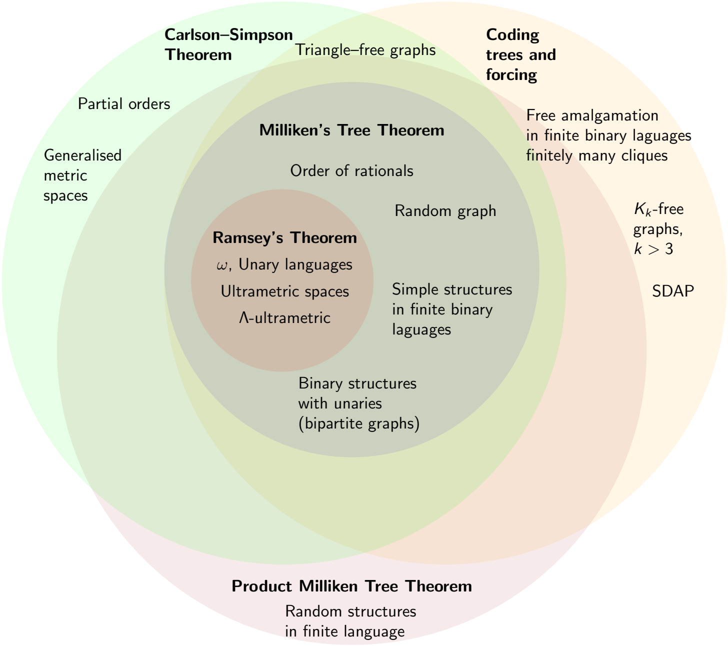
Known big Ramsey results by proof techniques



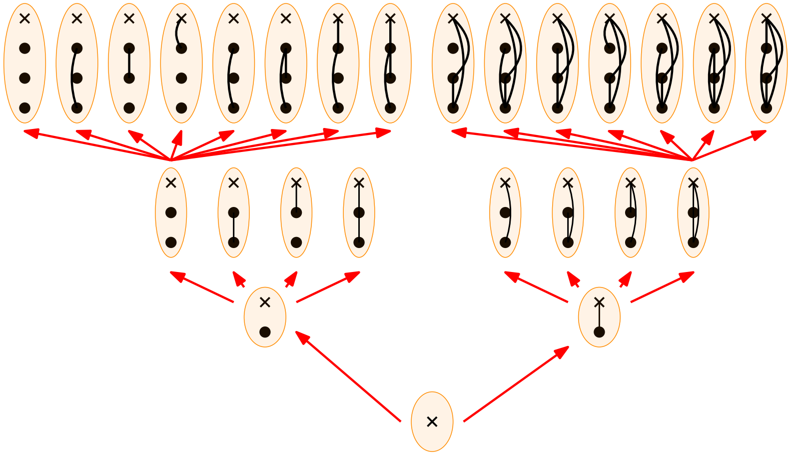
Known big Ramsey results by proof techniques



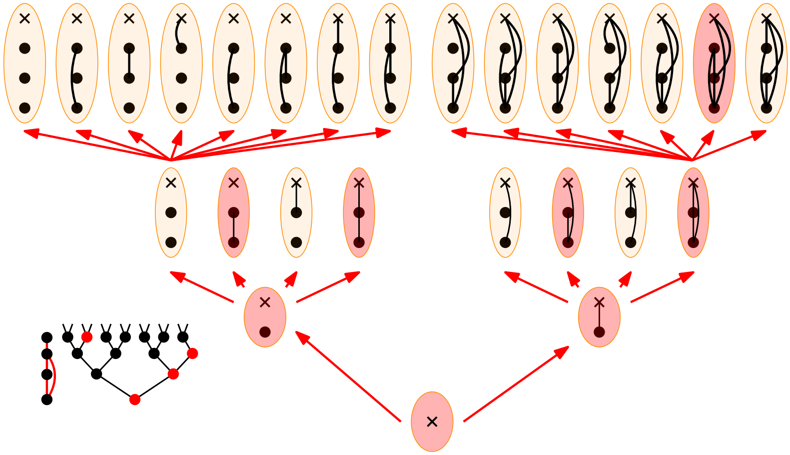
Known big Ramsey results by proof techniques



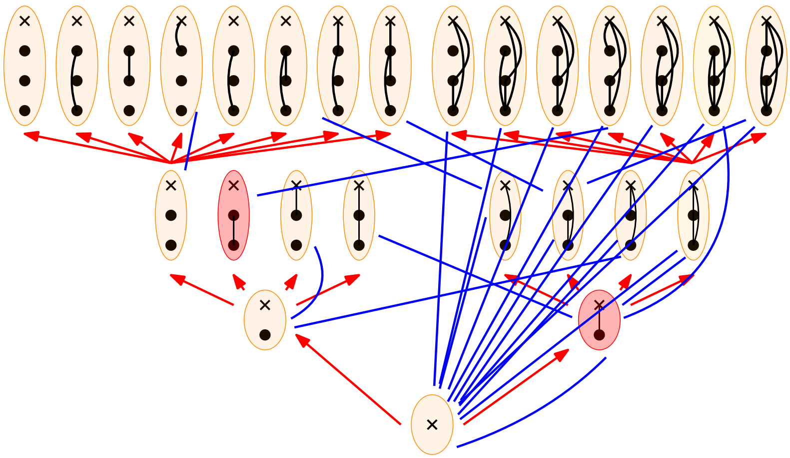
All enumerations tree



All enumerations tree



All enumerations tree



$\mathcal{S}$-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:

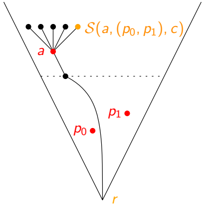
$\mathcal{S}$-trees

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, 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, 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 $c = d$.

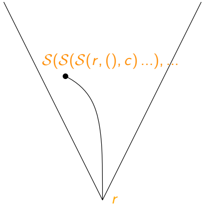
$\mathcal{S}$-trees

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, 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, 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)

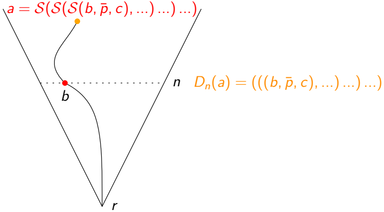
Given an $\mathcal{S}$-tree $(T, \preceq, \Sigma, \mathcal{S})$, we call a term α an $\mathcal{S}$-term if and only if $\alpha \in T$, or $\alpha = (\beta, (\gamma_0, \gamma_1, \dots, \gamma_{n-1}), c)$ where $n \in \omega$, all of $\beta, \gamma_0, \gamma_1 \dots \gamma_{n-1}$ are $\mathcal{S}$-terms 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 $a = \mathcal{S}(b, (p_0, \dots, p_{n-1}), c)$ such that $\ell(a) > n$, we let

$$\mathcal{D}_n(a) = (\mathcal{D}_n(b), (\mathcal{D}_n(p_0), \mathcal{D}_n(p_1), \dots, \mathcal{D}_n(p_{n-1})), c).$$



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:

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

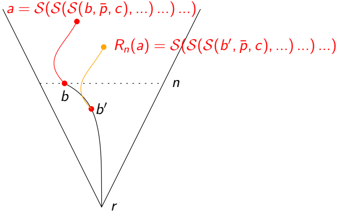
Manipulating nodes

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.

Example ($R_1(101) = 11$)

$$\begin{aligned} \mathcal{D}_2(101) &= (10, (), 1), \\ r_1(10) &= 10|_1 = 1, \\ r_1(\mathcal{D}_2(101)) &= r_1((10, (), 1)) = (r_1(10), (), 1) = (1, (), 1) = \mathcal{D}_1(11). \end{aligned}$$

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:

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

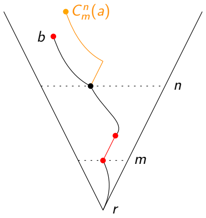
Manipulating nodes

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 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.

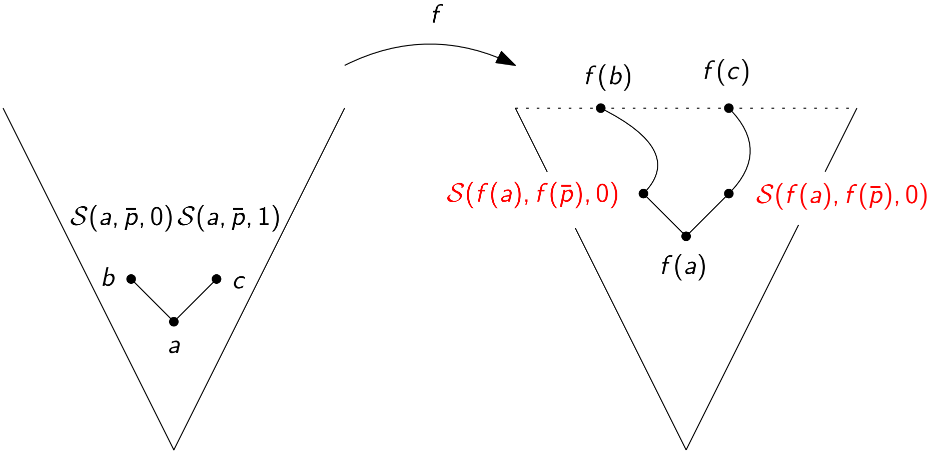


Definition (Shape-preserving functions)

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. Pigeonhole:

Proof outline (5 pages)

1. Pigeonhole:
   1. One-dimensional pigeonhole: either by application of the Hales-Jewett theorem or using ultrafilters

Proof outline (5 pages)

1. Pigeonhole:
   1. One-dimensional pigeonhole: either by application of the Hales-Jewett theorem or using ultrafilters
   2. Infinite-dimensional pigeonhole: combinatorial forcing inspired by proof by Karagianlis

Proof outline (5 pages)

1. Pigeonhole:
   1. One-dimensional pigeonhole: either by application of the Hales-Jewett theorem or using ultrafilters
   2. Infinite-dimensional pigeonhole: combinatorial forcing inspired by proof by Karagianlis
2. Coloring subtrees of a given finite size:
   Analogous fussion argument as in the proof of Milliken's tree theorem

Proof outline (5 pages)

1. Pigeonhole:
   1. One-dimensional pigeonhole: either by application of the Hales-Jewett theorem or using ultrafilters
   2. Infinite-dimensional pigeonhole: combinatorial forcing inspired by proof by Karagianlis
2. Coloring subtrees of a given finite size:
   Analogous fussion argument as in the proof of Milliken's tree theorem

Application to free amalgamation classes

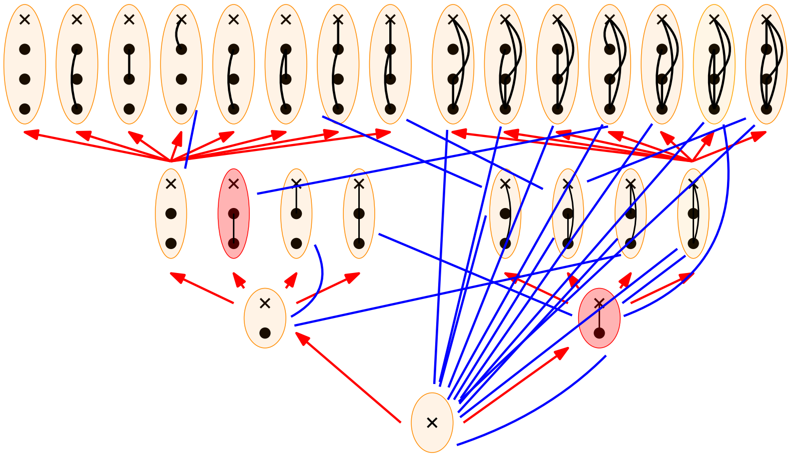
We want to give non-forcing proof of:

Theorem (Zucker 2020+)

Let L be a finite binary language and $\mathcal{F}$ a finite family of irreducible L-structures. Then every countable universal $\mathcal{F}$-free structure has finite big Ramsey degrees.

We fix family $\mathcal{F}$. Examples are for $\mathcal{F} = \{K_4\}$.

All enumerations tree



Constructing all enumeration tree

Definition (Type)

Type of level n is an $\mathcal{F}$-free L-structure $\mathbf{A}$ 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.

Constructing all enumeration tree

Definition (Type)

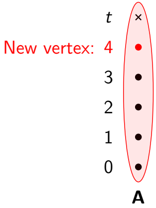
Type of level n is an $\mathcal{F}$-free L-structure $\mathbf{A}$ with vertices $\{0, 1, \dots, n-1, t\}$, where t is the type vertex.

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.



Constructing all enumeration tree

Definition (Type)

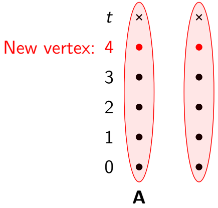
Type of level n is an $\mathcal{F}$-free L-structure $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

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.



Constructing all enumeration tree

Definition (Type)

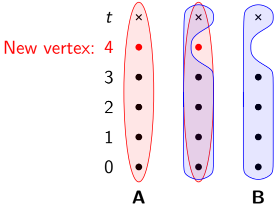
Type of level n is an $\mathcal{F}$-free L-structure $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

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.



Constructing all enumeration tree

Definition (Type)

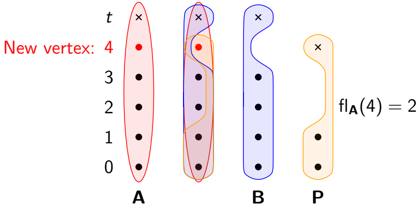
Type of level n is an $\mathcal{F}$-free L-structure $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

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.



Constructing all enumeration tree

Definition (Type)

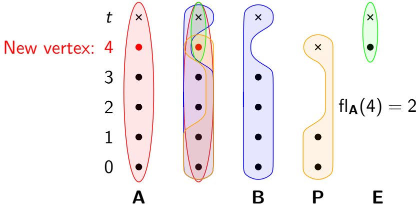
Type of level n is an $\mathcal{F}$-free L-structure $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

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.



Constructing all enumeration tree

Definition (Type)

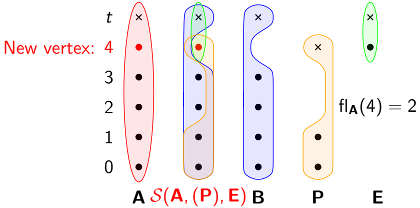
Type of level n is an $\mathcal{F}$-free L-structure $\mathbf{A}$ with vertices $\{0, 1, \ldots, n-1, t\}$, where t is the type vertex.

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:

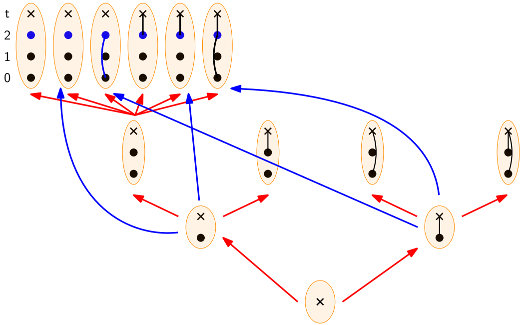
1. $\mathrm{fl}_{\mathbf{a}}(i) < i$.
2. whenever $i < j$ forms an edge of $\mathbf{A}$ then $\mathrm{fl}_{\mathbf{A}}(j) > i$.

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



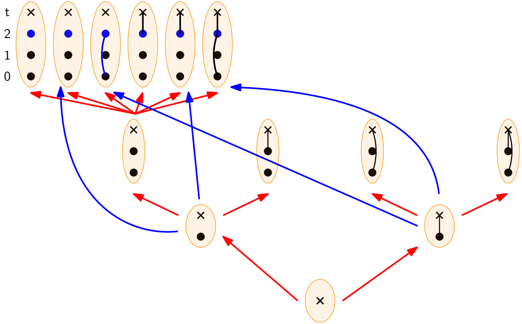
Non-forcing proof of Zucker's theorem

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



Non-forcing proof of Zucker's theorem

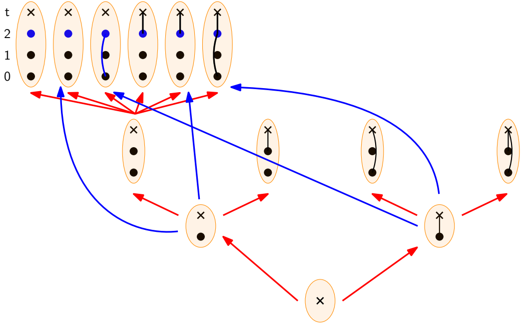
1. Build an $\mathcal{S}$-tree of levelled types:



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

Non-forcing proof of Zucker's theorem

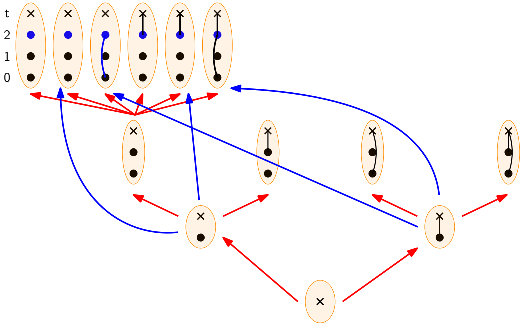
1. Build an $\mathcal{S}$-tree of levelled types:



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

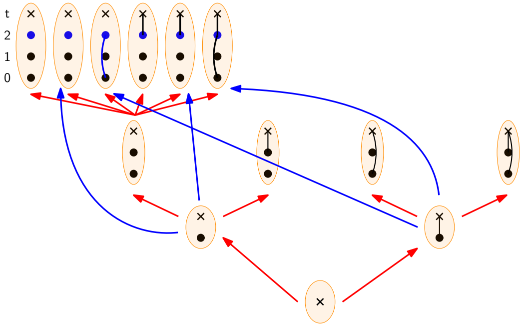
1. Build an $\mathcal{S}$-tree of levelled types:



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)

More general result

Theorem

Let L be a finite language consisting of unary and binary symbols, and let $\mathcal{K}$ be a hereditary class of finite structures and $k \geq 2$. Assume that every countable structure $\mathbf{A}$ *has a completion to $\mathcal{K}$ provided that every induced cycle in* $\mathbf{A}$ *(seen as a substructure) has a completion in $\mathcal{K}$ and every irreducible substructure of* $\mathbf{A}$ *of k embeds into $\mathcal{K}$. Then $\mathcal{K}$ has a Fraïssé limit with finite big Ramsey degrees.*

This result can be used to analyze all Cherlin's catalogues of binary homogeneous structures except for those described by infinitely many forbidden cliques (Henson graphs).

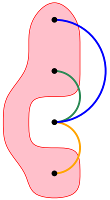
More general result

Theorem

Let L be a finite language consisting of unary and binary symbols, and let $\mathcal{K}$ be a hereditary class of finite structures and $k \geq 2$. Assume that every countable structure $\mathbf{A}$ *has a completion to $\mathcal{K}$ provided that every induced cycle in* $\mathbf{A}$ *(seen as a substructure) has a completion in $\mathcal{K}$ and every irreducible substructure of* $\mathbf{A}$ *of k embeds into $\mathcal{K}$. Then $\mathcal{K}$ has a Fraïssé limit with finite big Ramsey degrees.*

This result can be used to analyze all Cherlin's catalogues of binary homogeneous structures except for those described by infinitely many forbidden cliques (Henson graphs).

Open problem: Dpes class of all finite structures omitting the following substructure have finite big Ramsey degrees?



Thank you for the attention

1. D. Devlin: Some partition theorems and ultrafilters on ω, PhD thesis, Dartmouth College, 1979. See also: S. Todorcevic: Introduction to Ramsey Spaces.
2. C. Laflamme, N. Sauer, V. Vuksanovic: Canonical partitions of universal structures, Combinatorica 26 (2) (2006), 183–205.
3. N. Dobrinen, The Ramsey theory of the universal homogeneous triangle-free graph, Journal of Mathematical Logic 2020.
4. N. Dobrinen, The Ramsey Theory of Henson graphs, arXiv:1901.06660 (2019).
5. A. Zucker, Big Ramsey degrees and topological dynamics, Groups Geom. Dyn., 2018.
6. N. Dobrinen: The Ramsey theory of the universal homogeneous triangle-free graph Part II: Exact big Ramsey degrees, arXiv:2009.01985.
7. R. Coulson, N. Dobrinen, R. Patel: The Substructure Disjoint Amalgamation Property implies big Ramsey structures, arXiv:2010.02034.
8. J.H.: Big Ramsey degrees using parameter spaces, arXiv:2009.00967.
9. M. Balko, D. Chodounský, N. Dobrinen, J.H., M. Konečný, L. Vena, A. Zucker: Exact big Ramsey degrees via coding trees, arXiv:2110.08409 (2021).
10. M. Balko, D. Chodounský, J.H., M. Konečným J. Nešetřil, L. Vena: Big Ramsey degrees and forbidden cycles, Extended Abstracts EuroComb 2021.
11. M. Balko, D. Chodounský, N. Dobrinen, J.H., M. Konečný, J. Nešetřil, L. Vena, A. Zucker: Big Ramsey degrees of the generic partial order, Extended Abstracts EuroComb 2021.
12. M. Balko, D. Chodounský, J.H., M. Konečný, L. Vena: Big Ramsey degrees of 3-uniform hypergraphs are finite, Combintorica (2022).