A-Computable Graphs

Tyler Markkanen

Springfield College

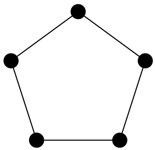
NERDS 8.0 – Assumption College
October 17, 2015

This is joint work with Matt Jura and Oscar Levin.

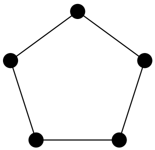
Domatic Partitions

Domatic 2-partition

Domatic 2-partition



No domatic 3-partition



Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

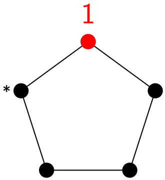
Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

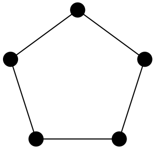
Domatic Partitions

Domatic 2-partition

Domatic 2-partition

No domatic 3-partition





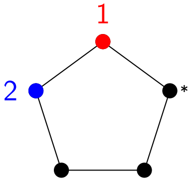
Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

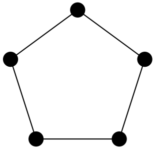
Domatic Partitions

Domatic 2-partition

Domatic 2-partition



No domatic 3-partition



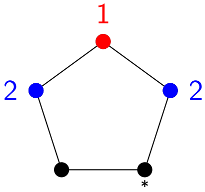
Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

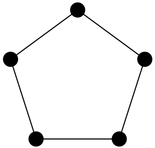
Domatic Partitions

Domatic 2-partition

Domatic 2-partition



No domatic 3-partition



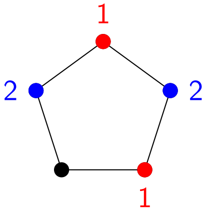
Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

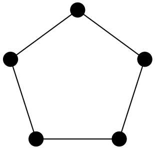
Domatic Partitions

Domatic 2-partition

Domatic 2-partition



No domatic 3-partition



Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

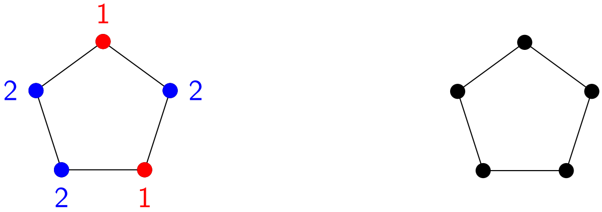
Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition

Domatic 2-partition

No domatic 3-partition



Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

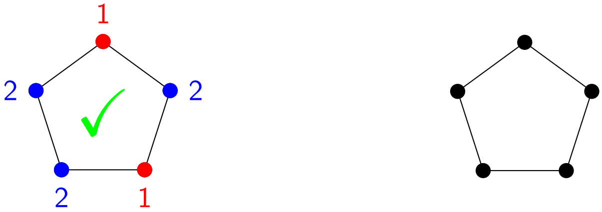
Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition

Domatic 2-partition

No domatic 3-partition



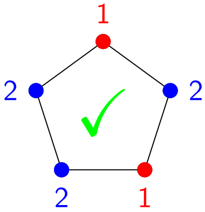
Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

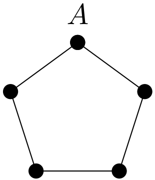
Domatic Partitions

Domatic 2-partition

Domatic 2-partition



No domatic 3-partition



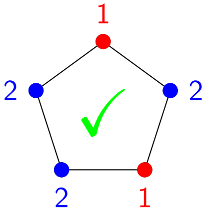
Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

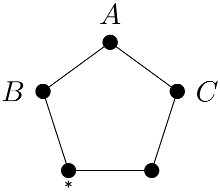
Domatic Partitions

Domatic 2-partition

Domatic 2-partition



No domatic 3-partition



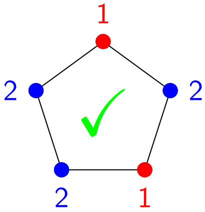
Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

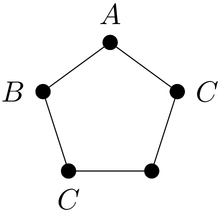
Domatic Partitions

Domatic 2-partition

Domatic 2-partition



No domatic 3-partition



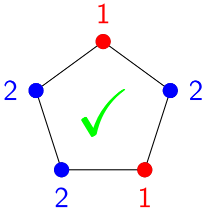
Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

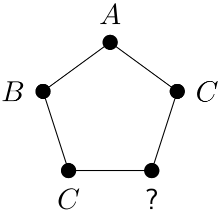
Domatic Partitions

Domatic 2-partition

Domatic 2-partition



No domatic 3-partition



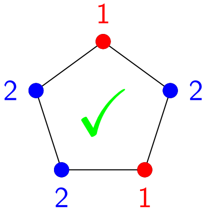
Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

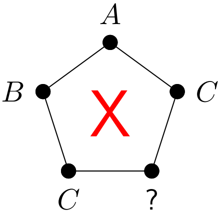
Domatic Partitions

Domatic 2-partition

Domatic 2-partition



No domatic 3-partition



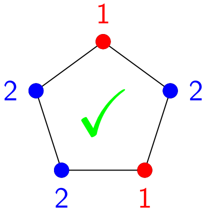
Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

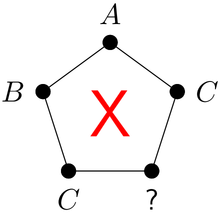
Domatic Partitions

Domatic 2-partition

Domatic 2-partition



No domatic 3-partition



$$d(G) = 2$$

Domatic number of a graph G:
$d(G) =$ the max n s.t. G has a domatic n-partition

Computable domatic number of a graph G:
$d^c(G) =$ the max n s.t. G has a computable domatic n-partition

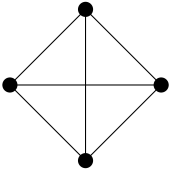
Computable Graphs

Example 1

There is a graph $G = (V, E)$ that is computable (i.e., V and E are computable sets) with the property that $d(G) = 3$ but $d^c(G) < 3$.

Let $\varphi_0, \varphi_1, \varphi_2, \ldots$ list all partial computable functions $\mathbb{N} \to \mathbb{N}$.

The gadget of φ_e:



$\rightarrow$

Springing the trap:

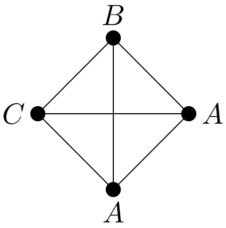
Computable Graphs

Example 1

There is a graph $G = (V, E)$ that is computable (i.e., V and E are computable sets) with the property that $d(G) = 3$ but $d^c(G) < 3$.

Let $\varphi_0, \varphi_1, \varphi_2, \ldots$ list all partial computable functions $\mathbb{N} \to \mathbb{N}$.

The gadget of φ_e:



$\rightarrow$

Springing the trap:

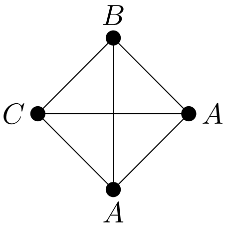
Computable Graphs

Example 1

There is a graph $G = (V, E)$ that is computable (i.e., V and E are computable sets) with the property that $d(G) = 3$ but $d^c(G) < 3$.

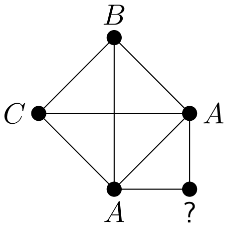
Let $\varphi_0, \varphi_1, \varphi_2, \ldots$ list all partial computable functions $\mathbb{N} \to \mathbb{N}$.

The gadget of φ_e:



$\rightarrow$

Springing the trap:



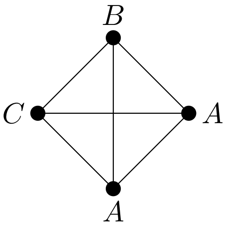
Computable Graphs

Example 1

There is a graph $G = (V, E)$ that is computable (i.e., V and E are computable sets) with the property that $d(G) = 3$ but $d^c(G) < 3$.

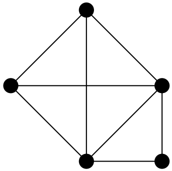
Let $\varphi_0, \varphi_1, \varphi_2, \ldots$ list all partial computable functions $\mathbb{N} \to \mathbb{N}$.

The gadget of φ_e:



$\rightarrow$

Springing the trap:



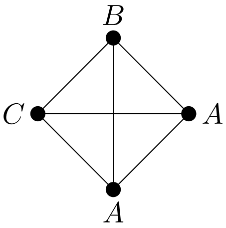
Computable Graphs

Example 1

There is a graph $G = (V, E)$ that is computable (i.e., V and E are computable sets) with the property that $d(G) = 3$ but $d^c(G) < 3$.

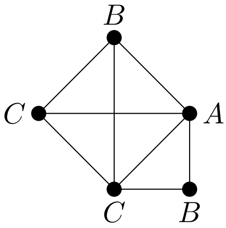
Let $\varphi_0, \varphi_1, \varphi_2, \ldots$ list all partial computable functions $\mathbb{N} \to \mathbb{N}$.

The gadget of φ_e:



$\rightarrow$

Springing the trap:



Highly Computable Graphs

Definition 1

- A graph G is **locally finite** if every vertex has finite degree (i.e., has finitely many neighbors).
- Given a locally finite computable graph $G = (V, E)$, let N_G denote the **neighborhood function** of G, which, on input $v \in V$, outputs the (code for) the set of neighbors of v. We say that G is A-**computable** if $N_G \leq_T A$.

Conjecture 1

For all $n \geq 2$, every highly computable (i.e., $\emptyset$-computable) graph that has a domatic n-partition also has a computable domatic $(n-1)$-partition.

Highly Computable Graphs

Definition 1

- A graph G is **locally finite** if every vertex has finite degree (i.e., has finitely many neighbors).
- Given a locally finite computable graph $G = (V, E)$, let N_G denote the **neighborhood function** of G, which, on input $v \in V$, outputs the (code for) the set of neighbors of v. We say that G is A-**computable** if $N_G \leq_T A$.

Conjecture 1

For all $n \geq 2$, every highly computable (i.e., $\emptyset$-computable) graph that has a domatic n-partition also has a computable domatic $(n-1)$-partition.

Highly Computable Graphs

Definition 1

- A graph G is **locally finite** if every vertex has finite degree (i.e., has finitely many neighbors).
- Given a locally finite computable graph $G = (V, E)$, let N_G denote the **neighborhood function** of G, which, on input $v \in V$, outputs the (code for) the set of neighbors of v. We say that G is A-**computable** if $N_G \leq_T A$.

Conjecture 1

For all $n \geq 2$, every highly computable (i.e., $\emptyset$-computable) graph that has a domatic n-partition also has a computable domatic $(n-1)$-partition.

Colorings

Theorem 1 (D. Bean)

There is a computable graph that has a finite coloring but no finite computable coloring.

Theorem 2 (J. Schmerl; H.G. Carstens and P. Päppinghaus)

Every highly computable graph that has an n-coloring has a computable $(2n-1)$-coloring.

Theorem 3 (W. Gasarch and A. Lee)

For any noncomputable c.e. set A, there is an A-computable graph that has a 2-coloring but no finite computable coloring.

Colorings

Theorem 1 (D. Bean)

There is a computable graph that has a finite coloring but no finite computable coloring.

Theorem 2 (J. Schmerl; H.G. Carstens and P. Päppinghaus)

Every highly computable graph that has an n-coloring has a computable $(2n-1)$-coloring.

Theorem 3 (W. Gasarch and A. Lee)

For any noncomputable c.e. set A, there is an A-computable graph that has a 2-coloring but no finite computable coloring.

Colorings

Theorem 1 (D. Bean)

There is a computable graph that has a finite coloring but no finite computable coloring.

Theorem 2 (J. Schmerl; H.G. Carstens and P. Päppinghaus)

Every highly computable graph that has an n-coloring has a computable $(2n-1)$-coloring.

Theorem 3 (W. Gasarch and A. Lee)

For any noncomputable c.e. set A, there is an A-computable graph that has a 2-coloring but no finite computable coloring.

Theorem 4

For any $n \geq 1$ and any noncomputable c.e. set A, there is an A-computable graph G such that $d(G) = n$ but $d^c(G) < 3$.

Setup for $n = 3$:

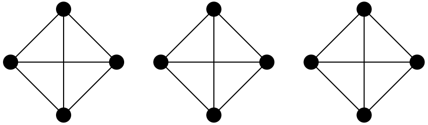
Gadgets for φ_e:

- Let L_i^e be the i-th gadget, $c_i^e = \max\{v : v \in L_i^e\}$, and $A_s = \{a_0, \ldots, a_s\}$ be a computable enumeration of A.
- We say that L_i^e **requires attention** if φ_e has converged on all of L_i^e so as to give L_i^e a domatic 3-partition.
- In this event, we additionally say L_i^e **deserves attention** if $a_s \leq c_i^e$. If this is the case, "spring the trap" for L_i^e.

Theorem 4

For any $n \geq 1$ and any noncomputable c.e. set A, there is an A-computable graph G such that $d(G) = n$ but $d^c(G) < 3$.

Setup for $n = 3$:

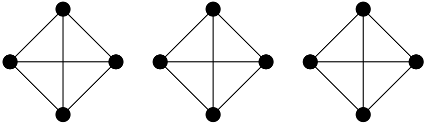
Gadgets for φ_e:  $\cdots$

- Let L_i^e be the i-th gadget, $c_i^e = \max\{v : v \in L_i^e\}$, and $A_s = \{a_0, \ldots, a_s\}$ be a computable enumeration of A.
- We say that L_i^e **requires attention** if φ_e has converged on all of L_i^e so as to give L_i^e a domatic 3-partition.
- In this event, we additionally say L_i^e **deserves attention** if $a_s \leq c_i^e$. If this is the case, "spring the trap" for L_i^e.

Theorem 4

For any $n \geq 1$ and any noncomputable c.e. set A, there is an A-computable graph G such that $d(G) = n$ but $d^c(G) < 3$.

Setup for $n = 3$:

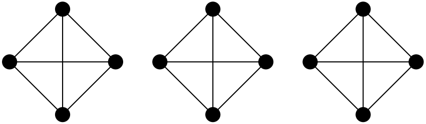
Gadgets for φ_e:  $\cdots$

- Let L_i^e be the i-th gadget, $c_i^e = \max\{v : v \in L_i^e\}$, and $A_s = \{a_0, \ldots, a_s\}$ be a computable enumeration of A.
- We say that L_i^e **requires attention** if φ_e has converged on all of L_i^e so as to give L_i^e a domatic 3-partition.
- In this event, we additionally say L_i^e **deserves attention** if $a_s \leq c_i^e$. If this is the case, "spring the trap" for L_i^e.

Theorem 4

For any $n \geq 1$ and any noncomputable c.e. set A, there is an A-computable graph G such that $d(G) = n$ but $d^c(G) < 3$.

Setup for $n = 3$:

Gadgets for φ_e:  $\cdots$

- Let L_i^e be the i-th gadget, $c_i^e = \max\{v : v \in L_i^e\}$, and $A_s = \{a_0, \ldots, a_s\}$ be a computable enumeration of A.
- We say that L_i^e **requires attention** if φ_e has converged on all of L_i^e so as to give L_i^e a domatic 3-partition.
- In this event, we additionally say L_i^e **deserves attention** if $a_s \leq c_i^e$. If this is the case, "spring the trap" for L_i^e.

Proving $N_G \leq_T A$

- Fix $v \in V$, and run the construction of G until we find e and i such that $v \in L_i^e$.
- Use A to find a stage t beyond which L_i^e will never change (whether or not its trap has sprung).
  - Indeed, run G out to the stages at which elements $x \leq c_i^e$ enter A, to determine if L_i^e deserves attention.
  - If L_i^e does not deserve attention by the last such stage, it never will afterward.
- So we will know all of the neighbors of each vertex in L_i^e (and, in particular, v) by t.

Proving $N_G \leq_T A$

- Fix $v \in V$, and run the construction of G until we find e and i such that $v \in L_i^e$.
- Use A to find a stage t beyond which L_i^e will never change (whether or not its trap has sprung).
  - Indeed, run G out to the stages at which elements $x \leq c_i^e$ enter A, to determine if L_i^e deserves attention.
  - If L_i^e does not deserve attention by the last such stage, it never will afterward.
- So we will know all of the neighbors of each vertex in L_i^e (and, in particular, v) by t.

Proving $N_G \leq_T A$

- Fix $v \in V$, and run the construction of G until we find e and i such that $v \in L_i^e$.
- Use A to find a stage t beyond which L_i^e will never change (whether or not its trap has sprung).
  - Indeed, run G out to the stages at which elements $x \leq c_i^e$ enter A, to determine if L_i^e deserves attention.
  - If L_i^e does not deserve attention by the last such stage, it never will afterward.
- So we will know all of the neighbors of each vertex in L_i^e (and, in particular, v) by t.

Proving $N_G \leq_T A$

- Fix $v \in V$, and run the construction of G until we find e and i such that $v \in L_i^e$.
- Use A to find a stage t beyond which L_i^e will never change (whether or not its trap has sprung).
  - Indeed, run G out to the stages at which elements $x \leq c_i^e$ enter A, to determine if L_i^e deserves attention.
  - If L_i^e does not deserve attention by the last such stage, it never will afterward.
- So we will know all of the neighbors of each vertex in L_i^e (and, in particular, v) by t.

Proving $d^c(G) < 3$

- Assume φ_e is a domatic 3-partition of G.
- Then we claim A is computable, a contradiction.
  - Indeed, let $n \in \mathbb{N}$, and run G until an L_i^e appears such that $c_i^e \geq n$.
  - Find the first stage t beyond this point such that $\varphi_{e,t}$ converges on L_i^e (which is unsprung).
  - Since L_i^e now requires attention but will never deserve it (by assumption), $A \upharpoonright c_i^e = A_t$.
  - So $n \in A \iff n \in A_t$ by our choice of c_i^e.

Proving $d^c(G) < 3$

- Assume φ_e is a domatic 3-partition of G.
- Then we claim A is computable, a contradiction.
  - Indeed, let $n \in \mathbb{N}$, and run G until an L_i^e appears such that $c_i^e \geq n$.
  - Find the first stage t beyond this point such that $\varphi_{e,t}$ converges on L_i^e (which is unsprung).
  - Since L_i^e now requires attention but will never deserve it (by assumption), $A \upharpoonright c_i^e = A_t$.
  - So $n \in A \iff n \in A_t$ by our choice of c_i^e.

Proving $d^c(G) < 3$

- Assume φ_e is a domatic 3-partition of G.
- Then we claim A is computable, a contradiction.
  - Indeed, let $n \in \mathbb{N}$, and run G until an L_i^e appears such that $c_i^e \geq n$.
  - Find the first stage t beyond this point such that $\varphi_{e,t}$ converges on L_i^e (which is unsprung).
  - Since L_i^e now requires attention but will never deserve it (by assumption), $A \upharpoonright c_i^e = A_t$.
  - So $n \in A \iff n \in A_t$ by our choice of c_i^e.

Proving $d^c(G) < 3$

- Assume φ_e is a domatic 3-partition of G.
- Then we claim A is computable, a contradiction.
  - Indeed, let $n \in \mathbb{N}$, and run G until an L_i^e appears such that $c_i^e \geq n$.
  - Find the first stage t beyond this point such that $\varphi_{e,t}$ converges on L_i^e (which is unsprung).
  - Since L_i^e now requires attention but will never deserve it (by assumption), $A \upharpoonright c_i^e = A_t$.
  - So $n \in A \iff n \in A_t$ by our choice of c_i^e.

Proving $d^c(G) < 3$

- Assume φ_e is a domatic 3-partition of G.
- Then we claim A is computable, a contradiction.
  - Indeed, let $n \in \mathbb{N}$, and run G until an L_i^e appears such that $c_i^e \geq n$.
  - Find the first stage t beyond this point such that $\varphi_{e,t}$ converges on L_i^e (which is unsprung).
  - Since L_i^e now requires attention but will never deserve it (by assumption), $A \upharpoonright c_i^e = A_t$.
  - So $n \in A \iff n \in A_t$ by our choice of c_i^e.

Euler Paths

Definition 2

- An **Euler path** of a graph $G = (V, E)$ is a sequence $v_0, v_1, \ldots \in V$ such that $v_i v_{i+1} \in E$ for all i and each edge in G appears exactly once in the sequence.
- A **computable Euler path** is a computable function f such that $f(n) = v_n$ for all $n \in \mathbb{N}$.

Theorem 5

For any noncomputable c.e. set A, there is an A-computable graph that has an Euler path but no computable Euler path.

Euler Paths

Definition 2

- An **Euler path** of a graph $G = (V, E)$ is a sequence $v_0, v_1, \ldots \in V$ such that $v_i v_{i+1} \in E$ for all i and each edge in G appears exactly once in the sequence.
- A **computable Euler path** is a computable function f such that $f(n) = v_n$ for all $n \in \mathbb{N}$.

Theorem 5

For any noncomputable c.e. set A, there is an A-computable graph that has an Euler path but no computable Euler path.

Euler Paths

Definition 2

- An **Euler path** of a graph $G = (V, E)$ is a sequence $v_0, v_1, \ldots \in V$ such that $v_i v_{i+1} \in E$ for all i and each edge in G appears exactly once in the sequence.
- A **computable Euler path** is a computable function f such that $f(n) = v_n$ for all $n \in \mathbb{N}$.

Theorem 5

For any noncomputable c.e. set A, there is an A-computable graph that has an Euler path but no computable Euler path.

Counterexample to a Generalization of Gasarch and Lee

Theorem 6

There is a noncomputable Δ_2^0 set A such that every finitely colorable A-computable graph has a finite computable coloring.

Requirements:

$\mathcal{P}_e$: $A \neq \varphi_e$

$\mathcal{R}_{\langle e,i,n \rangle}$: If ψ_i is an A-computable graph, via Φ_e^A, that has an n-coloring, then it has a finite computable coloring.

Order the requirements as: $\mathcal{P}_0 \prec \mathcal{R}_0 \prec \mathcal{P}_1 \prec \mathcal{R}_1 \prec \cdots$, where lower requirements in the ordering have higher priority.

Counterexample to a Generalization of Gasarch and Lee

Theorem 6

There is a noncomputable Δ^0_2 set A such that every finitely colorable A-computable graph has a finite computable coloring.

Requirements:

$\mathcal{P}_e$: $A \neq \varphi_e$

$\mathcal{R}_{\langle e,i,n\rangle}$: If ψ_i is an A-computable graph, via Φ_e^A, that has an n-coloring, then it has a finite computable coloring.

Order the requirements as: $\mathcal{P}_0 \prec \mathcal{R}_0 \prec \mathcal{P}_1 \prec \mathcal{R}_1 \prec \cdots$, where lower requirements in the ordering have higher priority.

Counterexample to a Generalization of Gasarch and Lee

Theorem 6

There is a noncomputable Δ_2^0 set A such that every finitely colorable A-computable graph has a finite computable coloring.

Requirements:

$\mathcal{P}_e$: $A \neq \varphi_e$

$\mathcal{R}_{\langle e,i,n\rangle}$: If ψ_i is an A-computable graph, via Φ_e^A, that has an n-coloring, then it has a finite computable coloring.

Order the requirements as: $\mathcal{P}_0 \prec \mathcal{R}_0 \prec \mathcal{P}_1 \prec \mathcal{R}_1 \prec \cdots$, where lower requirements in the ordering have higher priority.

Strategy for $\mathcal{P}_e$

$\mathcal{P}_e$: $A \neq \varphi_e$

- Pick an unused $x \in \mathbb{N}$ as a witness, and wait for $\varphi_e(x)\downarrow$.
  - If $\varphi_e(x)\downarrow = 0$, put x into A, and *issue restraint* on A up to x (i.e., prevent lower priority requirements from changing the membership in A of any $y \leq x$).
  - If $\varphi_e(x)\downarrow \neq 0$, do nothing.
- Given $\mathcal{Q} \prec \mathcal{P}_e$, if $\mathcal{Q}$ removes x from A after $\mathcal{P}_e$ put it in or if, at the time $\mathcal{P}_e$ is putting it in, $\mathcal{Q}$ has issued its own restraint above x, then we say $\mathcal{Q}$ *injures* $\mathcal{P}_e$. In this case, restart $\mathcal{P}_e$ with a new witness.

Strategy for $\mathcal{P}_e$

$\mathcal{P}_e$: $A \neq \varphi_e$

- Pick an unused $x \in \mathbb{N}$ as a witness, and wait for $\varphi_e(x)\downarrow$.
  - If $\varphi_e(x)\downarrow = 0$, put x into A, and *issue restraint* on A up to x (i.e., prevent lower priority requirements from changing the membership in A of any $y \leq x$).
  - If $\varphi_e(x)\downarrow \neq 0$, do nothing.
- Given $\mathcal{Q} \prec \mathcal{P}_e$, if $\mathcal{Q}$ removes x from A after $\mathcal{P}_e$ put it in or if, at the time $\mathcal{P}_e$ is putting it in, $\mathcal{Q}$ has issued its own restraint above x, then we say $\mathcal{Q}$ *injures* $\mathcal{P}_e$. In this case, restart $\mathcal{P}_e$ with a new witness.

Strategy for $\mathcal{P}_e$

$\mathcal{P}_e$: $A \neq \varphi_e$

- Pick an unused $x \in \mathbb{N}$ as a witness, and wait for $\varphi_e(x)\downarrow$.
  - If $\varphi_e(x)\downarrow = 0$, put x into A, and *issue restraint* on A up to x (i.e., prevent lower priority requirements from changing the membership in A of any $y \leq x$).
  - If $\varphi_e(x)\downarrow \neq 0$, do nothing.
- Given $\mathcal{Q} \prec \mathcal{P}_e$, if $\mathcal{Q}$ removes x from A after $\mathcal{P}_e$ put it in or if, at the time $\mathcal{P}_e$ is putting it in, $\mathcal{Q}$ has issued its own restraint above x, then we say $\mathcal{Q}$ *injures* $\mathcal{P}_e$. In this case, restart $\mathcal{P}_e$ with a new witness.

Strategy for $\mathcal{R}_{\langle e,i,n\rangle}$

$\mathcal{R}_{\langle e,i,n\rangle}$: $[\Phi_e^A = N_{\psi_i}$ & $\chi(\psi_i) \leq n] \implies \chi^c(\psi_i) < \infty$.

- Initially let $V_0 = \emptyset$, and V_t be the set of vertices seen by the end of stage $t-1$ of the strategy. At the beginning of stage t, put vertex t into V_t to ensure $\{0, \ldots, t\} \subseteq V_t$.
- Compute the set $N_{t,s}(v) = \{u \in V_t \cup \Phi_e^{A_s}(v) : \psi_i(u,v)\}$ for all $v \in V_t$, where s is the current stage of the entire construction.
- Let U_t be the set of all uncolored vertices of $V_t \cup \bigcup_{v \in V_t} N_{t,s}(v)$.

Strategy for $\mathcal{R}_{\langle e,i,n\rangle}$

$\mathcal{R}_{\langle e,i,n\rangle}$: $[\Phi_e^A = N_{\psi_i} \ \& \ \chi(\psi_i) \leq n] \implies \chi^c(\psi_i) < \infty$.

- Initially let $V_0 = \emptyset$, and V_t be the set of vertices seen by the end of stage $t-1$ of the strategy. At the beginning of stage t, put vertex t into V_t to ensure $\{0, \ldots, t\} \subseteq V_t$.
- Compute the set $N_{t,s}(v) = \{u \in V_t \cup \Phi_e^{A_s}(v) : \psi_i(u,v)\}$ for all $v \in V_t$, where s is the current stage of the entire construction.
- Let U_t be the set of all uncolored vertices of $V_t \cup \bigcup_{v \in V_t} N_{t,s}(v)$.

Strategy for $\mathcal{R}_{\langle e,i,n\rangle}$

$\mathcal{R}_{\langle e,i,n\rangle}$: $[\Phi_e^A = N_{\psi_i} \ \&\ \chi(\psi_i) \leq n] \implies \chi^c(\psi_i) < \infty$.

- Initially let $V_0 = \emptyset$, and V_t be the set of vertices seen by the end of stage $t-1$ of the strategy. At the beginning of stage t, put vertex t into V_t to ensure $\{0, \ldots, t\} \subseteq V_t$.
- Compute the set $N_{t,s}(v) = \{u \in V_t \cup \Phi_e^{A_s}(v) : \psi_i(u, v)\}$ for all $v \in V_t$, where s is the current stage of the entire construction.
- Let U_t be the set of all uncolored vertices of $V_t \cup \bigcup_{v \in V_t} N_{t,s}(v)$.

Strategy for $\mathcal{R}_{\langle e,i,n \rangle}$

$\mathcal{R}_{\langle e,i,n \rangle}$: $[\Phi_e^A = N_{\psi_i} \ \& \ \chi(\psi_i) \leq n] \implies \chi^c(\psi_i) < \infty$.

- Initially let $V_0 = \emptyset$, and V_t be the set of vertices seen by the end of stage $t-1$ of the strategy. At the beginning of stage t, put vertex t into V_t to ensure $\{0, \ldots, t\} \subseteq V_t$.
- Compute the set $N_{t,s}(v) = \{u \in V_t \cup \Phi_e^{A_s}(v) : \psi_i(u,v)\}$ for all $v \in V_t$, where s is the current stage of the entire construction.
- Let U_t be the set of all uncolored vertices of $V_t \cup \bigcup_{v \in V_t} N_{t,s}(v)$.

Strategy for $\mathcal{R}_{\langle e,i,n\rangle}$

$\mathcal{R}_{\langle e,i,n\rangle}$: $[\Phi_e^A = N_{\psi_i}$ & $\chi(\psi_i) \leq n] \implies \chi^c(\psi_i) < \infty$.

- Initially let $V_0 = \emptyset$, and V_t be the set of vertices seen by the end of stage $t-1$ of the strategy. At the beginning of stage t, put vertex t into V_t to ensure $\{0, \ldots, t\} \subseteq V_t$.
- Compute the set $N_{t,s}(v) = \{u \in V_t \cup \Phi_e^{A_s}(v) : \psi_i(u,v)\}$ for all $v \in V_t$, where s is the current stage of the entire construction.
- Let U_t be the set of all uncolored vertices of $V_t \cup \bigcup_{v \in V_t} N_{t,s}(v)$.

Strategy for $\mathcal{R}_{\langle e,i,n\rangle}$ (Cont'd)

- Color U_t with $\{1, \ldots, n\}$ or $\{n+1, \ldots, 2n\}$ alternatively (i.e., if we used the 1st set last time, then use the 2nd set this time, and vice versa).
    - If this coloring procedure is impossible, then there must be $u \in U_t$ adjacent to a previously colored $v \in V_t$. Since u is uncolored, it was absent from an earlier version of the neighborhood of v. So *rewind* A back to an earlier version $A_{s'}$ that computed the "wrong" neighborhood, and restrain A up to the use of $\Phi_e^{A_{s'}}$.
    - If a higher priority requirement prevents us from rewinding A, then color U_t with an *online* procedure (i.e., use colors beyond $2n$ as needed).
- Let $V_{t+1} = V_t \cup U_t$.

Strategy for $\mathcal{R}_{\langle e,i,n\rangle}$ (Cont'd)

- Color U_t with $\{1, \ldots, n\}$ or $\{n+1, \ldots, 2n\}$ alternatively (i.e., if we used the 1st set last time, then use the 2nd set this time, and vice versa).
  - If this coloring procedure is impossible, then there must be $u \in U_t$ adjacent to a previously colored $v \in V_t$. Since u is uncolored, it was absent from an earlier version of the neighborhood of v. So *rewind* A back to an earlier version $A_{s'}$ that computed the "wrong" neighborhood, and restrain A up to the use of $\Phi_e^{A_{s'}}$.
  - If a higher priority requirement prevents us from rewinding A, then color U_t with an *online* procedure (i.e., use colors beyond $2n$ as needed).
- Let $V_{t+1} = V_t \cup U_t$.

Strategy for $\mathcal{R}_{\langle e,i,n \rangle}$ (Cont'd)

- Color U_t with $\{1, \ldots, n\}$ or $\{n+1, \ldots, 2n\}$ alternatively (i.e., if we used the 1st set last time, then use the 2nd set this time, and vice versa).
  - If this coloring procedure is impossible, then there must be $u \in U_t$ adjacent to a previously colored $v \in V_t$. Since u is uncolored, it was absent from an earlier version of the neighborhood of v. So *rewind* A back to an earlier version $A_{s'}$ that computed the "wrong" neighborhood, and restrain A up to the use of $\Phi_e^{A_{s'}}$.
  - If a higher priority requirement prevents us from rewinding A, then color U_t with an *online* procedure (i.e., use colors beyond $2n$ as needed).
- Let $V_{t+1} = V_t \cup U_t$.

Strategy for $\mathcal{R}_{\langle e,i,n\rangle}$ (Cont'd)

- Color U_t with $\{1, \ldots, n\}$ or $\{n+1, \ldots, 2n\}$ alternatively (i.e., if we used the 1st set last time, then use the 2nd set this time, and vice versa).
  - If this coloring procedure is impossible, then there must be $u \in U_t$ adjacent to a previously colored $v \in V_t$. Since u is uncolored, it was absent from an earlier version of the neighborhood of v. So *rewind* A back to an earlier version $A_{s'}$ that computed the "wrong" neighborhood, and restrain A up to the use of $\Phi_e^{A_{s'}}$.
  - If a higher priority requirement prevents us from rewinding A, then color U_t with an *online* procedure (i.e., use colors beyond $2n$ as needed).
- Let $V_{t+1} = V_t \cup U_t$.

Strategy for $\mathcal{R}_{\langle e,i,n\rangle}$ (Cont'd)

- Color U_t with $\{1,\ldots,n\}$ or $\{n+1,\ldots,2n\}$ alternatively (i.e., if we used the 1st set last time, then use the 2nd set this time, and vice versa).
  - If this coloring procedure is impossible, then there must be $u \in U_t$ adjacent to a previously colored $v \in V_t$. Since u is uncolored, it was absent from an earlier version of the neighborhood of v. So *rewind* A back to an earlier version $A_{s'}$ that computed the "wrong" neighborhood, and restrain A up to the use of $\Phi_e^{A_{s'}}$.
  - If a higher priority requirement prevents us from rewinding A, then color U_t with an *online* procedure (i.e., use colors beyond $2n$ as needed).
- Let $V_{t+1} = V_t \cup U_t$.

Strategy for $\mathcal{R}_{\langle e,i,n \rangle}$ (Cont'd)

- Color U_t with $\{1, \ldots, n\}$ or $\{n+1, \ldots, 2n\}$ alternatively (i.e., if we used the 1st set last time, then use the 2nd set this time, and vice versa).
  - If this coloring procedure is impossible, then there must be $u \in U_t$ adjacent to a previously colored $v \in V_t$. Since u is uncolored, it was absent from an earlier version of the neighborhood of v. So *rewind* A back to an earlier version $A_{s'}$ that computed the "wrong" neighborhood, and restrain A up to the use of $\Phi_e^{A_{s'}}$.
  - If a higher priority requirement prevents us from rewinding A, then color U_t with an *online* procedure (i.e., use colors beyond $2n$ as needed).
- Let $V_{t+1} = V_t \cup U_t$.

Definition 3

A Δ_2^0 set A is **low for graph neighborhood (l.f.g.n.)** if every A-computable graph is highly computable.

Corollary to Gasarch and Lee

No noncomputable c.e. set is l.f.g.n.

Theorem 7

There is a noncomputable Δ_2^0 set A that is l.f.g.n.

Below is an alternative method for showing the existence of A:

Theorem 8

For every Δ_2^0 set A and A-computable graph G, there is a c.e. set $B \leq_T A$ such that G is B-computable.

Definition 3

A Δ_2^0 set A is **low for graph neighborhood (l.f.g.n.)** if every A-computable graph is highly computable.

Corollary to Gasarch and Lee

No noncomputable c.e. set is l.f.g.n.

Theorem 7

There is a noncomputable Δ_2^0 set A that is l.f.g.n.

Below is an alternative method for showing the existence of A:

Theorem 8

For every Δ_2^0 set A and A-computable graph G, there is a c.e. set $B \leq_T A$ such that G is B-computable.

Definition 3

A Δ_2^0 set A is **low for graph neighborhood (l.f.g.n.)** if every A-computable graph is highly computable.

Corollary to Gasarch and Lee

No noncomputable c.e. set is l.f.g.n.

Theorem 7

There is a noncomputable Δ_2^0 set A that is l.f.g.n.

Below is an alternative method for showing the existence of A:

Theorem 8

For every Δ_2^0 set A and A-computable graph G, there is a c.e. set $B \leq_T A$ such that G is B-computable.

Definition 3

A Δ_2^0 set A is **low for graph neighborhood (l.f.g.n.)** if every A-computable graph is highly computable.

Corollary to Gasarch and Lee

No noncomputable c.e. set is l.f.g.n.

Theorem 7

There is a noncomputable Δ_2^0 set A that is l.f.g.n.

Below is an alternative method for showing the existence of A:

Theorem 8

For every Δ_2^0 set A and A-computable graph G, there is a c.e. set $B \leq_T A$ such that G is B-computable.

Theorem 8

For every Δ_2^0 set A and A-computable graph G, there is a c.e. set $B \leq_T A$ such that G is B-computable.

Theorem 9

The following are equivalent for A noncomputable Δ_2^0.

1. *A is l.f.g.n.*
2. *Every c.e. set $B \leq_T A$ is computable.*
3. *Every A-computable graph that has a finite coloring has a finite computable coloring.*
4. *Every A-computable graph that has an Euler path has a computable Euler path.*

Thank you.

Dwight R. Bean.
Effective coloration.
J. Symbolic Logic, 41(2):469–480, 1976.

Dwight R. Bean.
Recursive Euler and Hamilton paths.
Proc. Amer. Math. Soc., 55(2):385–394, 1976.

W. Gasarch.
A survey of recursive combinatorics.
In *Handbook of recursive mathematics, Vol. 2*, volume 139 of *Stud. Logic Found. Math.*, pages 1041–1176. North-Holland, Amsterdam, 1998.

William I. Gasarch and Andrew C. Y. Lee.
On the finiteness of the recursive chromatic number.
Ann. Pure Appl. Logic, 93(1-3):73–81, 1998.
Computability theory.