Can Computers Do Math? An Introduction to Computability Theory and Effective Mathematics

Tyler Markkanen

Springfield College

CCSU Mathematics Department Colloquium
February 27, 2015

Joint with Matthew Jura and Oscar Levin.

Machines & Self-Reference

- Consider a machine M that prints out expressions made with five symbols:

$$\sim, \quad P, \quad N, \quad (, \quad \text{and} \quad).$$

- An **expression** is any non-empty finite string of symbols, e.g.,

$$N{\sim}(P, \qquad)P(((((((, \qquad \text{and} \qquad P(N(\sim)).$$

- For an expression X, a **sentence** is any expression of the form:

$$P(X), \quad PN(X), \quad {\sim}P(X), \quad \text{or} \quad {\sim}PN(X).$$

- We interpret the meaning of the symbols as follows.
  P: "is printable"
  $\sim$: "not"
  N: "the **norm** of" E.g., the norm of $P{\sim}$ is $P{\sim}(P{\sim})$.

Telling the Truth

Rule:
The machine M can only print *TRUE* sentences.

Example 1

1. If M prints $P(X)$, then X is printable. So M eventually prints X.
2. If M prints $\sim PN(X)$, then the norm of X, i.e., $X(X)$, is not printable. So M never prints $X(X)$.
3. If M prints X, then M does not necessarily print $P(X)$.

Question:
Can such an M print *ALL* true sentences?

You Can't Handle the Truth!

Can such an M print *ALL* true sentences?

No. The following sentence is true but M will not print it:

$$\sim PN(\sim PN)$$

Notice:

$$\sim PN(\sim PN) \text{ is true} \iff \sim PN(\sim PN) \text{ is not printable}$$

Sets of Natural Numbers

- The set of natural numbers: $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ — countable
- The subsets of $\mathbb{N}$: $\varnothing, \mathbb{N},$

$$\{0\}, \{1\}, \{2\}, \{3\}, \ldots,$$
$$\{0,1\}, \{0,2\}, \{1,2\}, \{0,3\}, \{1,3\}, \{2,3\}, \ldots,$$
$$\{0,1,2\}, \ldots,$$
$$\vdots$$
$$\{0,2,4,6,8,\ldots\}, \{1,3,5,7,9,\ldots\},$$
$$\{2,3,5,7,11,\ldots\}, \ldots$$
$$\vdots$$

— uncountably many

(We can't number ALL the subsets of $\mathbb{N}$.)

Sets of Natural Numbers

- The set of natural numbers: $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ — countable
- The subsets of $\mathbb{N}$: $\emptyset, \mathbb{N}$,

$$\{0\}, \{1\}, \{2\}, \{3\}, \ldots,$$
$$\{0,1\}, \{0,2\}, \{1,2\}, \{0,3\}, \{1,3\}, \{2,3\}, \ldots,$$
$$\{0,1,2\}, \ldots,$$
$$\vdots$$
$$\{0,2,4,6,8,\ldots\}, \{1,3,5,7,9,\ldots\},$$
$$\{2,3,5,7,11,\ldots\}, \ldots$$
$$\vdots$$

— uncountably many

(We can't number ALL the subsets of $\mathbb{N}$.)

Sets of Natural Numbers

- The set of natural numbers: $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ — countable
- The subsets of $\mathbb{N}$: $\varnothing, \mathbb{N},$

$$\begin{array}{l}\{0\}, \{1\}, \{2\}, \{3\}, \ldots, \\ \{0,1\}, \{0,2\}, \{1,2\}, \{0,3\}, \{1,3\}, \{2,3\}, \ldots, \\ \{0,1,2\}, \ldots, \\ \vdots \\ \{0,2,4,6,8,\ldots\}, \{1,3,5,7,9,\ldots\}, \\ \{2,3,5,7,11,\ldots\}, \ldots \\ \vdots\end{array}$$

— uncountably many

(We can't number ALL the subsets of $\mathbb{N}$.)

What Is a Computer Program?

Example 2 ($a + b$)

Input: a, b
Want **Output:** $a + b$
Program:

1. $s := a$
2. $i := 0$
3. IF $\{ \quad i \neq b$
   i. $s := s + 1$
   ii. $i := i + 1 \}$
4. Print s

A **(computer) program** is a machine with a finite list of steps (written from a finite alphabet) that takes in a natural number (the **input**), runs the steps on the input, and (if it stops running) prints out a natural number (the **output**).

What Is a Computer Program?

Example 2 ($a+b$)

Input: a, b

Want **Output:** $a+b$

Program:

1. $s := a$
2. $i := 0$
3. IF $\{$ $i \neq b$
   i. $s := s+1$
   ii. $i := i+1$ $\}$
4. Print s

A **(computer) program** is a machine with a finite list of steps (written from a finite alphabet) that takes in a natural number (the **input**), runs the steps on the input, and (if it stops running) prints out a natural number (the **output**).

What Is a Computer Program?

Example 2 ($a + b$)

Input: a, b
Want **Output:** $a + b$
Program:

1. $s := a$
2. $i := 0$
3. IF $\{$ $i \neq b$
   i. $s := s + 1$
   ii. $i := i + 1$ $\}$
4. Print s

A **(computer) program** is a machine with a finite list of steps (written from a finite alphabet) that takes in a natural number (the **input**), runs the steps on the input, and (if it stops running) prints out a natural number (the **output**).

What Is a Computer Program?

Example 2 ($a + b$)

Input: a, b

Want **Output:** $a + b$

Program:

1. $s := a$
2. $i := 0$
3. IF $\{$ $i \neq b$
   i. $s := s + 1$
   ii. $i := i + 1$ $\}$
4. Print s

A **(computer) program** is a machine with a finite list of steps (written from a finite alphabet) that takes in a natural number (the **input**), runs the steps on the input, and (if it stops running) prints out a natural number (the **output**).

Examples of Programs

A computer program is a partial function $f : \mathbb{N} \to \mathbb{N}$ (which may not have an output for some inputs).

Example 3 (Familiar programs)

$a + b$, $a \dot{-} b$, $a > b$, $a \cdot b$, a^b, $a | b$, EVEN(a), ODD(a)

Example 4 (A program where some inputs have no output)

Program Name: EVENstopODDdontstop

Input: a

Program:

1. If a is EVEN, Then Print 0.
2. If a is ODD, Then Go To Step 1.

Output: $\begin{cases} 0, & \text{if } a \text{ is even} \\ \uparrow \text{ (no output)}, & \text{if } a \text{ is odd} \end{cases}$

Examples of Programs

A computer program is a partial function $f : \mathbb{N} \to \mathbb{N}$ (which may not have an output for some inputs).

Example 3 (Familiar programs)

$a + b$, $a \dot{-} b$, $a > b$, $a \cdot b$, a^b, $a|b$, EVEN(a), ODD(a)

Example 4 (A program where some inputs have no output)

Program Name: EVENstopODDdontstop
Input: a
Program:

1. If a is EVEN, Then Print 0.
2. If a is ODD, Then Go To Step 1.

Output: $\begin{cases} 0, & \text{if } a \text{ is even} \\ \uparrow \text{ (no output)}, & \text{if } a \text{ is odd} \end{cases}$

Examples of Programs

A computer program is a partial function $f : \mathbb{N} \to \mathbb{N}$ (which may not have an output for some inputs).

Example 3 (Familiar programs)

$a + b$, $a \dot{-} b$, $a > b$, $a \cdot b$, a^b, $a|b$, EVEN(a), ODD(a)

Example 4 (A program where some inputs have no output)

Program Name: EVENstopODDdontstop

Input: a

Program:

1. If a is EVEN, Then Print 0.
2. If a is ODD, Then Go To Step 1.

Output: $\begin{cases} 0, & \text{if } a \text{ is even} \\ \uparrow \text{ (no output)}, & \text{if } a \text{ is odd} \end{cases}$

Examples of Programs

A computer program is a partial function $f : \mathbb{N} \to \mathbb{N}$ (which may not have an output for some inputs).

Example 3 (Familiar programs)

$a + b$, $a \dot{-} b$, $a > b$, $a \cdot b$, a^b, $a|b$, EVEN(a), ODD(a)

Example 4 (A program where some inputs have no output)

Program Name: EVENstopODDdontstop
Input: a
Program:

1. If a is EVEN, Then Print 0.
2. If a is ODD, Then Go To Step 1.

Output: $\begin{cases} 0, & \text{if } a \text{ is even} \\ \uparrow \text{ (no output)}, & \text{if } a \text{ is odd} \end{cases}$

Examples of Programs

A computer program is a partial function $f : \mathbb{N} \to \mathbb{N}$ (which may not have an output for some inputs).

Example 3 (Familiar programs)

$a + b$, $a \dot{-} b$, $a > b$, $a \cdot b$, a^b, $a|b$, EVEN(a), ODD(a)

Example 4 (A program where some inputs have no output)

Program Name: EVENstopODDdontstop
Input: a
Program:

1. If a is EVEN, Then Print 0.
2. If a is ODD, Then Go To Step 1.

Output: $\begin{cases} 0, & \text{if } a \text{ is even} \\ \uparrow \ \text{(no output)}, & \text{if } a \text{ is odd} \end{cases}$

Computable Functions and Sets

- Each program is a finite list of steps. So how many different programs are there?
- So we can number ALL of the programs:

  $\varphi_0, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \cdots$

Definition 1

- *We call these φ_e the* **partial computable functions**. *If φ_e is total (i.e., $\mathrm{dom}(\varphi_e) = \mathbb{N}$), we call it a* **computable function**.
- *$A \subseteq \mathbb{N}$ is called a* **computable set** *if*

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

is a computable function.

Computable Functions and Sets

- Each program is a finite list of steps. So how many different programs are there? Only countably many (the size of $\mathbb{N}$).
- So we can number ALL of the programs:

  $\varphi_0, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \cdots$

Definition 1

- *We call these* φ_e *the* **partial computable functions**. *If* φ_e *is total (i.e.,* $\mathrm{dom}(\varphi_e) = \mathbb{N}$*), we call it a* **computable function**.
- $A \subseteq \mathbb{N}$ *is called a* **computable set** *if*

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

is a computable function.

Computable Functions and Sets

- Each program is a finite list of steps. So how many different programs are there? Only countably many (the size of $\mathbb{N}$).
- So we can number ALL of the programs:

  $\varphi_0, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \cdots$

Definition 1

- *We call these* φ_e *the* **partial computable functions**. *If* φ_e *is total (i.e.,* $\mathrm{dom}(\varphi_e) = \mathbb{N}$*), we call it a* **computable function**.
- $A \subseteq \mathbb{N}$ *is called a* **computable set** *if*

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

is a computable function.

Computable Functions and Sets

- Each program is a finite list of steps. So how many different programs are there? Only countably many (the size of $\mathbb{N}$).
- So we can number ALL of the programs:

$\varphi_0, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \cdots$

Definition 1

- *We call these φ_e the* **partial computable functions**. *If φ_e is total (i.e., $\mathrm{dom}(\varphi_e) = \mathbb{N}$), we call it a* **computable function**.
- *$A \subseteq \mathbb{N}$ is called a* **computable set** *if*

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

is a computable function.

Computable Functions and Sets

- Each program is a finite list of steps. So how many different programs are there? Only countably many (the size of $\mathbb{N}$).
- So we can number ALL of the programs:

$\varphi_0, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \cdots$

Definition 1

- *We call these φ_e the* **partial computable functions**. *If φ_e is total (i.e., $\mathrm{dom}(\varphi_e) = \mathbb{N}$), we call it a* **computable function**.
- *$A \subseteq \mathbb{N}$ is called a* **computable set** *if*

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

is a computable function.

Examples of Computable Sets

Example 5

- $\varnothing$

  because $\chi_\varnothing(x) = \begin{cases} 1 & \text{if } x \in \varnothing \\ 0 & \text{if } x \notin \varnothing \end{cases} = 0 \quad \text{(for all } x\text{)}$

- $\mathbb{N}$

  because $\chi_\mathbb{N}(x) = \begin{cases} 1 & \text{if } x \in \mathbb{N} \\ 0 & \text{if } x \notin \mathbb{N} \end{cases} = 1 \quad \text{(for all } x\text{)}$

- A, where A is a finite set
- $E = \{x : x \text{ is even}\}$
- $O = \{x : x \text{ is odd}\}$
- $\overline{C}$, where C is a computable set
- $S = \{x : x \text{ is a perfect square}\}$
- $P = \{x : x \text{ is a prime number}\}$

Examples of Computable Sets

Example 5

- $\varnothing$

  because $\chi_{\varnothing}(x) = \begin{cases} 1 & \text{if } x \in \varnothing \\ 0 & \text{if } x \notin \varnothing \end{cases} = 0 \quad (\text{for all } x)$

- $\mathbb{N}$

  because $\chi_{\mathbb{N}}(x) = \begin{cases} 1 & \text{if } x \in \mathbb{N} \\ 0 & \text{if } x \notin \mathbb{N} \end{cases} = 1 \quad (\text{for all } x)$

- A, where A is a finite set
- $E = \{x : x \text{ is even}\}$
- $O = \{x : x \text{ is odd}\}$
- $\overline{C}$, where C is a computable set
- $S = \{x : x \text{ is a perfect square}\}$
- $P = \{x : x \text{ is a prime number}\}$

Examples of Computable Sets

Example 5

- $\varnothing$

  because $\chi_{\varnothing}(x) = \begin{cases} 1 & \text{if } x \in \varnothing \\ 0 & \text{if } x \notin \varnothing \end{cases} = 0 \quad \text{(for all } x\text{)}$

- $\mathbb{N}$

  because $\chi_{\mathbb{N}}(x) = \begin{cases} 1 & \text{if } x \in \mathbb{N} \\ 0 & \text{if } x \notin \mathbb{N} \end{cases} = 1 \quad \text{(for all } x\text{)}$

- A, where A is a finite set
- $E = \{x : x \text{ is even}\}$
- $O = \{x : x \text{ is odd}\}$
- $\overline{C}$, where C is a computable set
- $S = \{x : x \text{ is a perfect square}\}$
- $P = \{x : x \text{ is a prime number}\}$

Examples of Computable Sets

Example 5

- $\varnothing$

  because $\chi_{\varnothing}(x) = \begin{cases} 1 & \text{if } x \in \varnothing \\ 0 & \text{if } x \notin \varnothing \end{cases} = 0 \quad$ (for all x)

- $\mathbb{N}$

  because $\chi_{\mathbb{N}}(x) = \begin{cases} 1 & \text{if } x \in \mathbb{N} \\ 0 & \text{if } x \notin \mathbb{N} \end{cases} = 1 \quad$ (for all x)

- A, where A is a finite set
- $E = \{x : x \text{ is even}\}$
- $O = \{x : x \text{ is odd}\}$
- $\overline{C}$, where C is a computable set
- $S = \{x : x \text{ is a perfect square}\}$
- $P = \{x : x \text{ is a prime number}\}$

Examples of Computable Sets

Example 5

- $\varnothing$

  because $\chi_{\varnothing}(x) = \begin{cases} 1 & \text{if } x \in \varnothing \\ 0 & \text{if } x \notin \varnothing \end{cases} = 0 \quad \text{(for all } x\text{)}$

- $\mathbb{N}$

  because $\chi_{\mathbb{N}}(x) = \begin{cases} 1 & \text{if } x \in \mathbb{N} \\ 0 & \text{if } x \notin \mathbb{N} \end{cases} = 1 \quad \text{(for all } x\text{)}$

- A, where A is a finite set
- $E = \{x : x \text{ is even}\}$
- $O = \{x : x \text{ is odd}\}$
- $\overline{C}$, where C is a computable set
- $S = \{x : x \text{ is a perfect square}\}$
- $P = \{x : x \text{ is a prime number}\}$

Examples of Computable Sets

Example 5

- $\varnothing$

  because $\chi_{\varnothing}(x) = \begin{cases} 1 & \text{if } x \in \varnothing \\ 0 & \text{if } x \notin \varnothing \end{cases} = 0 \quad \text{(for all } x\text{)}$

- $\mathbb{N}$

  because $\chi_{\mathbb{N}}(x) = \begin{cases} 1 & \text{if } x \in \mathbb{N} \\ 0 & \text{if } x \notin \mathbb{N} \end{cases} = 1 \quad \text{(for all } x\text{)}$

- A, where A is a finite set
- $E = \{x : x \text{ is even}\}$
- $O = \{x : x \text{ is odd}\}$
- $\overline{C}$, where C is a computable set
- $S = \{x : x \text{ is a perfect square}\}$
- $P = \{x : x \text{ is a prime number}\}$

Examples of Computable Sets

Example 5

- $\varnothing$

  because $\chi_{\varnothing}(x) = \begin{cases} 1 & \text{if } x \in \varnothing \\ 0 & \text{if } x \notin \varnothing \end{cases} = 0 \quad (\text{for all } x)$

- $\mathbb{N}$

  because $\chi_{\mathbb{N}}(x) = \begin{cases} 1 & \text{if } x \in \mathbb{N} \\ 0 & \text{if } x \notin \mathbb{N} \end{cases} = 1 \quad (\text{for all } x)$

- A, where A is a finite set
- $E = \{x : x \text{ is even}\}$
- $O = \{x : x \text{ is odd}\}$
- $\overline{C}$, where C is a computable set
- $S = \{x : x \text{ is a perfect square}\}$
- $P = \{x : x \text{ is a prime number}\}$

Examples of Computable Sets

Example 5

- $\varnothing$

  because $\chi_{\varnothing}(x) = \begin{cases} 1 & \text{if } x \in \varnothing \\ 0 & \text{if } x \notin \varnothing \end{cases} = 0 \quad \text{(for all } x\text{)}$
- $\mathbb{N}$

  because $\chi_{\mathbb{N}}(x) = \begin{cases} 1 & \text{if } x \in \mathbb{N} \\ 0 & \text{if } x \notin \mathbb{N} \end{cases} = 1 \quad \text{(for all } x\text{)}$
- A, where A is a finite set
- $E = \{x : x \text{ is even}\}$
- $O = \{x : x \text{ is odd}\}$
- $\overline{C}$, where C is a computable set
- $S = \{x : x \text{ is a perfect square}\}$
- $P = \{x : x \text{ is a prime number}\}$

Examples of Computable Sets

Example 5

- $\varnothing$

  because $\chi_{\varnothing}(x) = \begin{cases} 1 & \text{if } x \in \varnothing \\ 0 & \text{if } x \notin \varnothing \end{cases} = 0 \quad \text{(for all } x\text{)}$

- $\mathbb{N}$

  because $\chi_{\mathbb{N}}(x) = \begin{cases} 1 & \text{if } x \in \mathbb{N} \\ 0 & \text{if } x \notin \mathbb{N} \end{cases} = 1 \quad \text{(for all } x\text{)}$

- A, where A is a finite set
- $E = \{x : x \text{ is even}\}$
- $O = \{x : x \text{ is odd}\}$
- $\overline{C}$, where C is a computable set
- $S = \{x : x \text{ is a perfect square}\}$
- $P = \{x : x \text{ is a prime number}\}$

Are There Non-Computable Sets?

- Are there any sets that are NOT computable?
- There are only many computable sets, but many subsets of $\mathbb{N}$.
- There are many non-computable sets!
- Can I see one?

Are There Non-Computable Sets?

- Are there any sets that are NOT computable?
- There are only many computable sets, but many subsets of $\mathbb{N}$.
- There are many non-computable sets!
- Can I see one?

Are There Non-Computable Sets?

- Are there any sets that are NOT computable?
- There are only *countably* many computable sets, but many subsets of $\mathbb{N}$.
- There are many non-computable sets!
- Can I see one?

Are There Non-Computable Sets?

- Are there any sets that are NOT computable?
- There are only *countably* many computable sets, but *uncountably* many subsets of $\mathbb{N}$.
- There are many non-computable sets!
- Can I see one?

Are There Non-Computable Sets?

- Are there any sets that are NOT computable?
- There are only *countably* many computable sets, but *uncountably* many subsets of $\mathbb{N}$.
- There are many non-computable sets!
- Can I see one?

Are There Non-Computable Sets?

- Are there any sets that are NOT computable?
- There are only *countably* many computable sets, but *uncountably* many subsets of $\mathbb{N}$.
- There are many non-computable sets!
- Can I see one?

A Not-Computable Set

Example 6

The **Halting Problem**:
$K = \{e : \varphi_e(e)\downarrow\}$

($\varphi_e(e)\downarrow$ means "φ_e on input e stops running")

What does K mean?

- Say EVENstopODDdontstop is φ_{12}, so $e = 12$. Does $\varphi_{12}(12)$ stop or not stop?
- K is "ALL the e such that $\varphi_e(e)$ stops running."

A Not-Computable Set

Example 6

The **Halting Problem**:
$K = \{e : \varphi_e(e)\downarrow\}$

($\varphi_e(e)\downarrow$ means "φ_e on input e stops running")

What does K mean?

- Say EVENstopODDdontstop is φ_{12}, so $e = 12$. Does $\varphi_{12}(12)$ stop or not stop?
- K is "ALL the e such that $\varphi_e(e)$ stops running."

A Not-Computable Set

Example 6

The **Halting Problem**:
$K = \{e : \varphi_e(e)\downarrow\}$

($\varphi_e(e)\downarrow$ means "φ_e on input e stops running")

What does K mean?

- Say EVENstopODDdontstop is φ_{12}, so $e = 12$. Does $\varphi_{12}(12)$ stop or not stop?
- K is "ALL the e such that $\varphi_e(e)$ stops running."

A Not-Computable Set

Example 6

The **Halting Problem**:
$K = \{e : \varphi_e(e)\downarrow\}$

($\varphi_e(e)\downarrow$ means "φ_e on input e stops running")

What does K mean?

- Say EVENstopODDdontstop is φ_{12}, so $e = 12$. Does $\varphi_{12}(12)$ stop or not stop?
- K is "ALL the e such that $\varphi_e(e)$ stops running."

A Not-Computable Set

Example 6

The **Halting Problem**:
$K = \{e : \varphi_e(e)\downarrow\}$

($\varphi_e(e)\downarrow$ means "φ_e on input e stops running")

What does K mean?

- Say EVENstopODDdontstop is φ_{12}, so $e = 12$. Does $\varphi_{12}(12)$ stop or not stop?
- K is "ALL the e such that $\varphi_e(e)$ stops running."

A Not-Computable Set

Example 6

The **Halting Problem**:
$K = \{e : \varphi_e(e)\downarrow\}$

($\varphi_e(e)\downarrow$ means "φ_e on input e stops running")

What does K mean?

- Say EVENstopODDdontstop is φ_{12}, so $e = 12$. Does $\varphi_{12}(12)$ stop or not stop? It stops running. So $12 \in K$.
- K is "ALL the e such that $\varphi_e(e)$ stops running."

A Not-Computable Set

Example 6

The **Halting Problem**:
$K = \{e : \varphi_e(e)\downarrow\}$

($\varphi_e(e)\downarrow$ means "φ_e on input e stops running")

What does K mean?

- Say EVENstopODDdontstop is φ_{12}, so $e = 12$. Does $\varphi_{12}(12)$ stop or not stop? It stops running. So $12 \in K$.
- K is "ALL the e such that $\varphi_e(e)$ stops running."

Why Is $K = \{e : \varphi_e(e)\downarrow\}$ Not Computable?

- Assume K were a computable set. That is, assume $\chi_K(x) = \begin{cases} 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$ is a computable function.
- **Claim:** The following function is also computable:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

Why? **Input:** x

Program:

If $\chi_K(x) = 1$, Then Print $\varphi_x(x) + 1$

If $\chi_K(x) = 0$, Then Print 0

On the other hand: $f \neq \varphi_e$ for any e because:

If $e \in K$, then $f(e) = \varphi_e(e) + 1 \neq \varphi_e(e)$.

If $e \notin K$, then $f(e) = 0 \neq \varphi_e(e)$ (because $\varphi_e(e)$ doesn't stop).

Why Is $K = \{e : \varphi_e(e)\downarrow\}$ Not Computable?

- Assume K were a computable set. That is, assume $\chi_K(x) = \begin{cases} 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$ is a computable function.
- **Claim:** The following function is also computable:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

Why? **Input:** x

Program:

If $\chi_K(x) = 1$, Then Print $\varphi_x(x) + 1$

If $\chi_K(x) = 0$, Then Print 0

On the other hand: $f \neq \varphi_e$ for any e because:

If $e \in K$, then $f(e) = \varphi_e(e) + 1 \neq \varphi_e(e)$.

If $e \notin K$, then $f(e) = 0 \neq \varphi_e(e)$ (because $\varphi_e(e)$ doesn't stop).

Why Is $K = \{e : \varphi_e(e)\downarrow\}$ Not Computable?

- Assume K were a computable set. That is, assume $\chi_K(x) = \begin{cases} 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$ is a computable function.
- **Claim:** The following function is also computable:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

Why? **Input:** x

Program:

If $\chi_K(x) = 1$, Then Print $\varphi_x(x) + 1$

If $\chi_K(x) = 0$, Then Print 0

On the other hand: $f \neq \varphi_e$ for any e because:

If $e \in K$, then $f(e) = \varphi_e(e) + 1 \neq \varphi_e(e)$.

If $e \notin K$, then $f(e) = 0 \neq \varphi_e(e)$ (because $\varphi_e(e)$ doesn't stop).

Why Is $K = \{e : \varphi_e(e)\downarrow\}$ Not Computable?

- Assume K were a computable set. That is, assume $\chi_K(x) = \begin{cases} 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$ is a computable function.
- **Claim:** The following function is also computable:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

Why? **Input:** x

Program:

If $\chi_K(x) = 1$, Then Print $\varphi_x(x) + 1$

If $\chi_K(x) = 0$, Then Print 0

On the other hand: $f \neq \varphi_e$ for any e because:

If $e \in K$, then $f(e) = \varphi_e(e) + 1 \neq \varphi_e(e)$.

If $e \notin K$, then $f(e) = 0 \neq \varphi_e(e)$ (because $\varphi_e(e)$ doesn't stop).

Why Is $K = \{e : \varphi_e(e)\downarrow\}$ Not Computable?

- Assume K were a computable set. That is, assume $\chi_K(x) = \begin{cases} 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$ is a computable function.
- **Claim:** The following function is also computable:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

Why? **Input:** x

Program:

If $\chi_K(x) = 1$, Then Print $\varphi_x(x) + 1$

If $\chi_K(x) = 0$, Then Print 0

On the other hand: $f \neq \varphi_e$ for any e because:

If $e \in K$, then $f(e) = \varphi_e(e) + 1 \neq \varphi_e(e)$.

If $e \notin K$, then $f(e) = 0 \neq \varphi_e(e)$ (because $\varphi_e(e)$ doesn't stop).

Why Is $K = \{e : \varphi_e(e)\downarrow\}$ Not Computable?

- Assume K were a computable set. That is, assume $\chi_K(x) = \begin{cases} 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$ is a computable function.
- **Claim:** The following function is also computable:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

Why? **Input:** x
Program:
If $\chi_K(x) = 1$, Then Print $\varphi_x(x) + 1$
If $\chi_K(x) = 0$, Then Print 0

On the other hand: $f \neq \varphi_e$ for any e because:
If $e \in K$, then $f(e) = \varphi_e(e) + 1 \neq \varphi_e(e)$.
If $e \notin K$, then $f(e) = 0 \neq \varphi_e(e)$ (because $\varphi_e(e)$ doesn't stop).

Why Is $K = \{e : \varphi_e(e)\downarrow\}$ Not Computable?

- Assume K were a computable set. That is, assume $\chi_K(x) = \begin{cases} 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$ is a computable function.
- **Claim:** The following function is also computable:

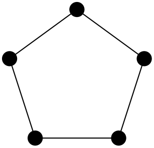
$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

Why? **Input:** x
Program:
If $\chi_K(x) = 1$, Then Print $\varphi_x(x) + 1$
If $\chi_K(x) = 0$, Then Print 0

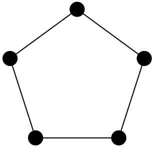
On the other hand: $f \neq \varphi_e$ for any e because:
If $e \in K$, then $f(e) = \varphi_e(e) + 1 \neq \varphi_e(e)$.
If $e \notin K$, then $f(e) = 0 \neq \varphi_e(e)$ (because $\varphi_e(e)$ doesn't stop).

Domatic Partitions

Domatic 2-partition



No domatic 3-partition

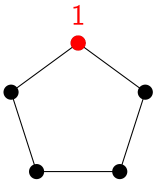


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

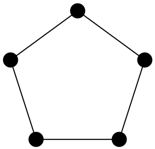
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



No domatic 3-partition

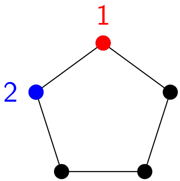


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

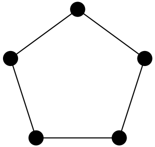
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



No domatic 3-partition

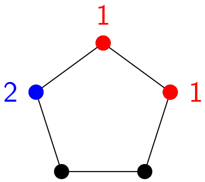


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

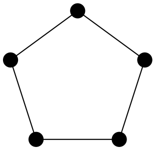
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



No domatic 3-partition

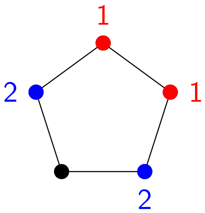


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

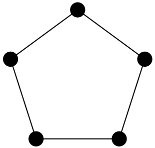
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



No domatic 3-partition

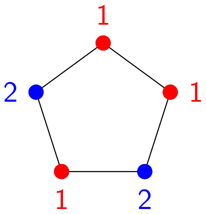


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

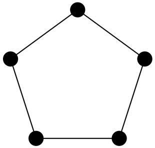
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



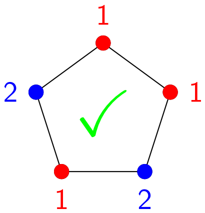
No domatic 3-partition

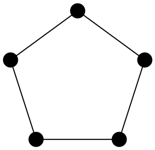


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

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

Domatic Partitions



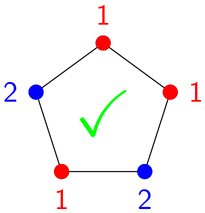


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

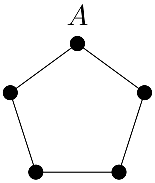
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



No domatic 3-partition

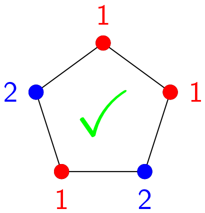


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

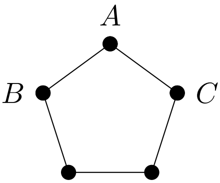
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



No domatic 3-partition

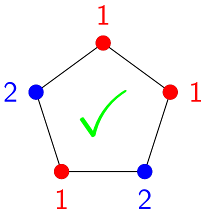


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

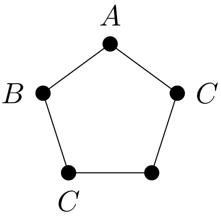
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



No domatic 3-partition

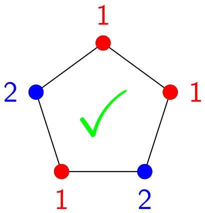


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

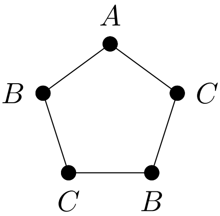
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



No domatic 3-partition

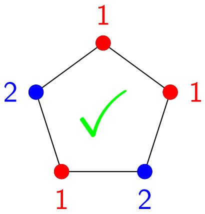


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

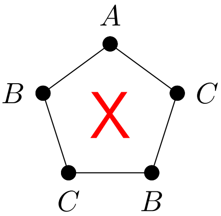
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



No domatic 3-partition

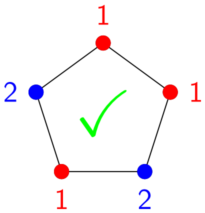


Domatic number of a graph G:
$d(G)$ = the max n s.t. G has a domatic n-partition

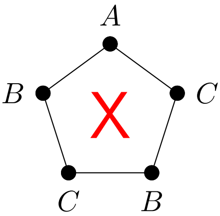
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Domatic Partitions

Domatic 2-partition



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

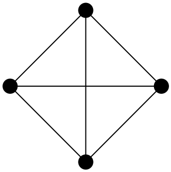
Computable domatic number of a graph G:
$d^c(G)$ = the max n s.t. G has a computable domatic n-partition

Separating the Domatic Numbers

Example 7

There is a computable graph G such that $d(G) = 3$ but $d^c(G) < 3$.

The gadget of φ_e:



$\rightarrow$

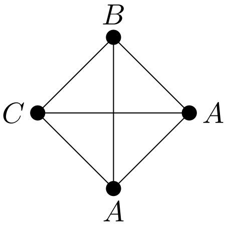
Springing the trap:

Separating the Domatic Numbers

Example 7

There is a computable graph G such that $d(G) = 3$ but $d^c(G) < 3$.

The gadget of φ_e:



→

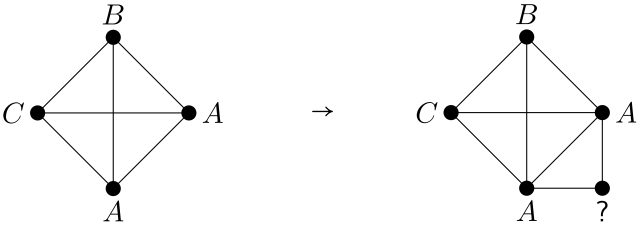
Springing the trap:

Separating the Domatic Numbers

Example 7

There is a computable graph G such that $d(G) = 3$ but $d^c(G) < 3$.

The gadget of φ_e: Springing the trap:

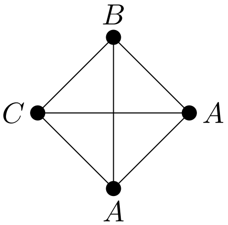


Separating the Domatic Numbers

Example 7

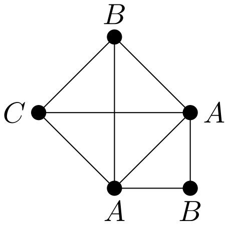
There is a computable graph G such that $d(G) = 3$ but $d^c(G) < 3$.

The gadget of φ_e:



→

Springing the trap:

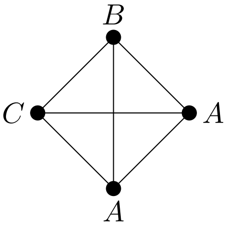


Separating the Domatic Numbers

Example 7

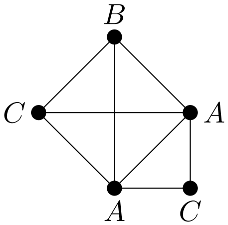
There is a computable graph G such that $d(G) = 3$ but $d^c(G) < 3$.

The gadget of φ_e:



→

Springing the trap:

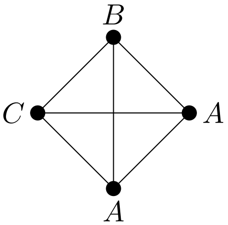


Separating the Domatic Numbers

Example 7

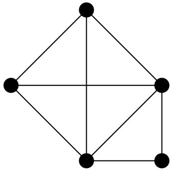
There is a computable graph G such that $d(G) = 3$ but $d^c(G) < 3$.

The gadget of φ_e:



→

Springing the trap:

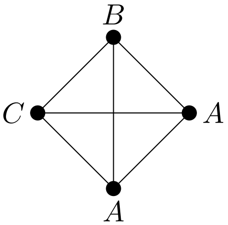


Separating the Domatic Numbers

Example 7

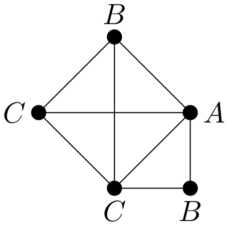
There is a computable graph G such that $d(G) = 3$ but $d^c(G) < 3$.

The gadget of φ_e:



$\rightarrow$

Springing the trap:



$d(G) - d^c(G)$ for Highly Computable Graphs G

Definition 2

A graph $G = (V, E)$ is **highly computable** *if V and E are computable sets and there is a computable function that, when given $v \in V$, outputs the degree of v (i.e., the number of vertices adjacent to v).*

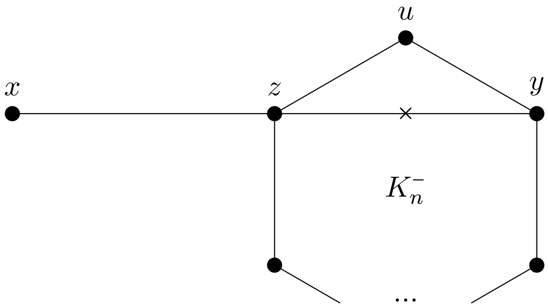
Theorem 1

For every $n \geq 3$, there is a highly computable graph G such that $d(G) = n$ and $d^c(G) = n - 1$.

The K_n^--Gadget

Theorem 1

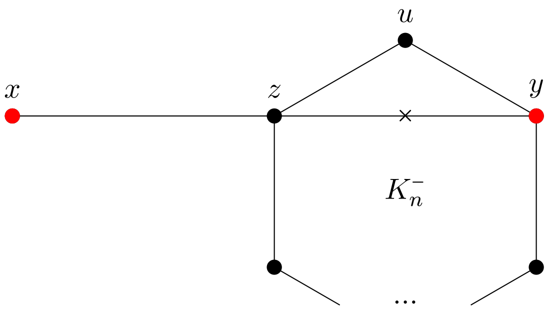
For every $n \geq 3$, there is a highly computable graph G such that $d(G) = n$ and $d^c(G) = n - 1$.



The K_n^--Gadget

Theorem 1

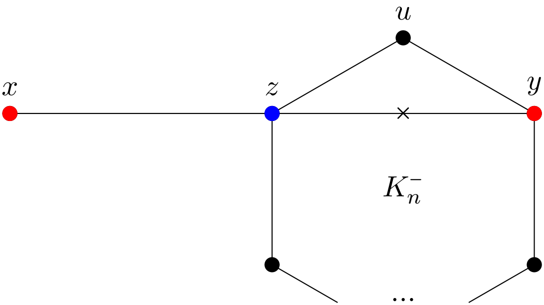
For every $n \geq 3$, there is a highly computable graph G such that $d(G) = n$ and $d^c(G) = n - 1$.



The K_n^--Gadget

Theorem 1

For every $n \geq 3$, there is a highly computable graph G such that $d(G) = n$ and $d^c(G) = n - 1$.



Total Domatic Partitions

Definition 3

1. *For any $n \geq 1$ and any graph $G = (V, E)$, a (computable) partition $p : V \to \{1, \ldots, n\}$ into n colors is a* **(computable) total domatic n-partition** *if the vertices adjacent to v use up all n colors (i.e., $(\forall v \in V)(\forall i \in \{1, \ldots, n\})(\exists u \in V)[uEv \wedge p(u) = i]$).*
2. *The* **(computable) total domatic number** *of a graph G, denoted by $d_t(G)$ (resp., $d_t^c(G)$) is the maximum n such that G has a (computable) total domatic n-partition.*

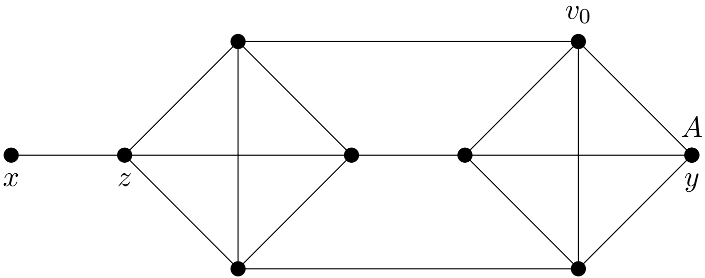
Theorem 2

For every $n \geq 3$, there is a highly computable graph G such that $d_t(G) = n$ and $d_t^c(G) = n - 1$.

The Double K_4-Gadget

Theorem 2

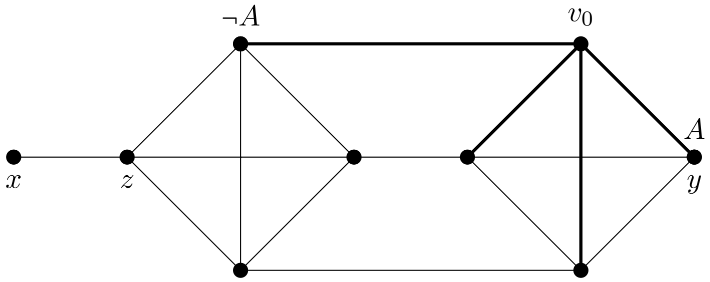
For every $n \geq 3$, there is a highly computable graph G such that $d_t(G) = n$ and $d_t^c(G) = n - 1$.



The Double K_4-Gadget

Theorem 2

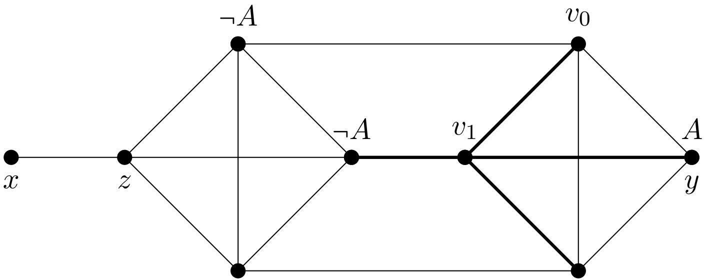
For every $n \geq 3$, there is a highly computable graph G such that $d_t(G) = n$ and $d_t^c(G) = n - 1$.



The Double K_4-Gadget

Theorem 2

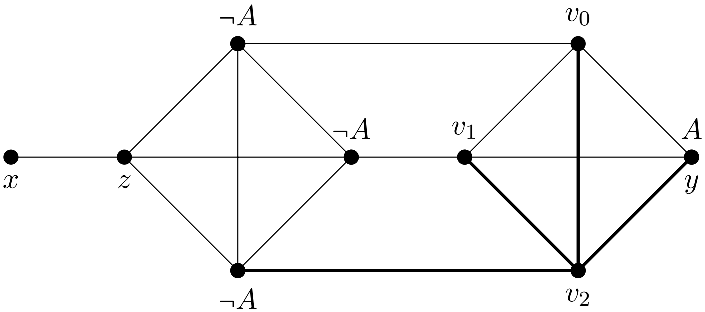
For every $n \geq 3$, there is a highly computable graph G such that $d_t(G) = n$ and $d_t^c(G) = n - 1$.



The Double K_4-Gadget

Theorem 2

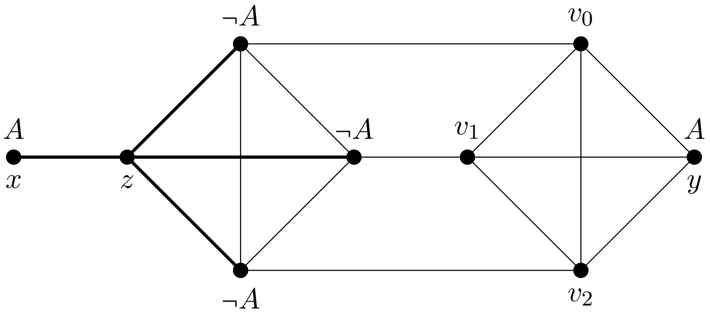
For every $n \geq 3$, there is a highly computable graph G such that $d_t(G) = n$ and $d_t^c(G) = n - 1$.



The Double K_4-Gadget

Theorem 2

For every $n \geq 3$, there is a highly computable graph G such that $d_t(G) = n$ and $d_t^c(G) = n - 1$.



Future Research

Conjecture 1

Every highly computable graph with a domatic 4-partition has a computable domatic 3-partition.

Theorem 3

Let G be a computable k-regular graph with $d(G) = k + 1$. Then G has a computable domatic n-partition for all n satisfying $2^n - 1 \leq k + 1$, in fact, $n^2 \leq k + 1$.

Thank you.

Matthew Jura, Oscar Levin, and Tyler Markkanen.
Domatic partitions of computable graphs.
Arch. Math. Logic, 53(1-2):137–155, 2014.

Robert I. Soare.
Computability Theorey and Applications [CTA].
under contract with Springer-Verlag, Berlin, 20??
(under revision).