Towards a fine structure of computabilities

Vers une structure fine des calculabilités

Fabien Givors

under the supervision of Grégory Lafitte and Bruno Durand

Université Montpellier II - CNRS, LIRMM

Montpellier

December 6th 2013

How did we get there?

How did we get there?

Quest n°1

- Explore the structure of recursively enumerable degrees
- Find *natural* objects with complex properties
- Get rid of technical proofs

How did we get there?

Quest n°1

- Explore the structure of recursively enumerable degrees
- Find *natural* objects with complex properties
- Get rid of technical proofs

Quest n°2

- Build models for fragments of computability
- Find out where and when the difficulty of computability arises
- Find a general framework for studying computabilities

1. classical computability

Church, Kleene, Rosser, Turing

Computable functions

Computable functions

Basic operations

$$\mathbf{o} : x \mapsto 0$$

$$\mathbf{s} : x \mapsto x + 1$$

$$\mathbf{cond} : x, y, a, b \mapsto \begin{cases} a & \text{if } x = y \\ b & \text{otherwise} \end{cases}$$

Computable functions

Basic operations

$$\mathbf{o} : x \mapsto 0$$

$$\mathbf{s} : x \mapsto x + 1$$

$$\mathbf{cond} : x, y, a, b \mapsto \begin{cases} a & \text{if } x = y \\ b & \text{otherwise} \end{cases}$$

Operations can be composed

$$\mathbf{s} \circ \mathbf{o} = x \mapsto 1$$

Computable functions

Basic operations

$\mathbf{o} : x \mapsto 0$

$\mathbf{s} : x \mapsto x + 1$

$\mathbf{cond} : x, y, a, b \mapsto \begin{cases} a & \text{if } x = y \\ b & \text{otherwise} \end{cases}$

Operations can be composed

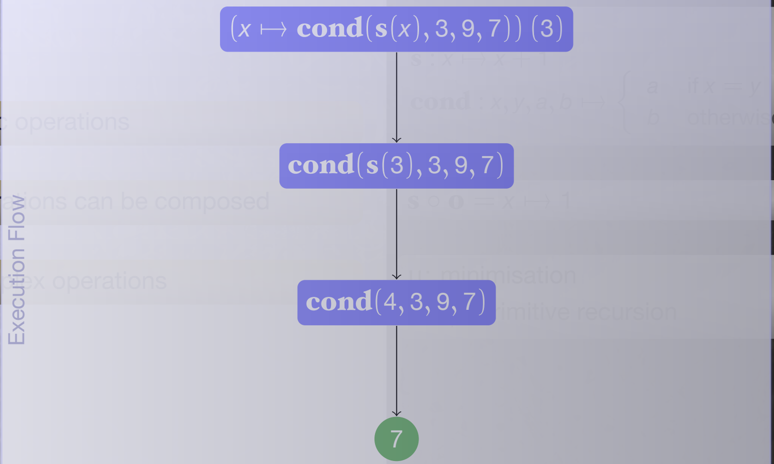
$\mathbf{s} \circ \mathbf{o} = x \mapsto 1$

Complex operations

μ: minimisation

$\mathbf{rec}_p$: primitive recursion

Computable functions



The more complex the machine, the more complex the flow.

Computable functions

Basic operations

$\mathbf{o} : x \mapsto 0$
$\mathbf{s} : x \mapsto x + 1$
$\mathbf{cond} : x, y, a, b \mapsto \begin{cases} a & \text{if } x = y \\ b & \text{otherwise} \end{cases}$

Operations can be composed

$\mathbf{s} \circ \mathbf{o} = x \mapsto 1$

Complex operations

μ: minimisation
$\mathbf{rec}_{\mathrm{p}}$: primitive recursion

Execution flow

Not necessarily finite

Notations

Enumeration of all the recursive functions, indexed by the natural integers:

$$(\varphi_e)_{e\in\omega}$$

Convergence of a function:

$$\varphi_{26}(13)\downarrow= 7$$

Divergence of a function:

$$\varphi_{34}(4)\uparrow$$

Equivalence of functions:

$$\varphi_{e_1}(x_1)\simeq\varphi_{e_2}(x_2)$$

if both computations diverge or converge to the same value.

Universality and total functions

Recursive universal function

$$\exists u, \varphi_u : (\langle e, x\rangle) \mapsto \varphi_e(x)$$

Universality and total functions

Recursive universal function

Functions cannot all be total

$$\exists u, \varphi_u : (\langle e, x\rangle) \mapsto \varphi_e(x)$$

$f : x \mapsto \mathbf{s} \circ \varphi_u(\langle x, x\rangle)$

$\exists e, f = \varphi_e$

$$\begin{aligned} f(e) &\cong \mathbf{s}(\varphi_u(\langle e, e\rangle)) \\ &\cong \varphi_e(e) + 1 \\ &\cong f(e) + 1 \end{aligned}$$

Canonical form and interpretation

Kleene's Normal Form

There exist an elementary function F and a recursive primitive predicate T such that:

$$\forall e, x,\ \varphi_u\left(\langle e, x\rangle\right) \cong \varphi_e(x) \cong F(\ \mu y.T(e, x, y))$$

- Only μ operator
- Function index
- Execution flow
- Input
- Checker

Canonical form and interpretation

Kleene's Normal Form

There exist an elementary function F and a recursive primitive predicate T such that:

$$\forall e, x,\ \varphi_u(\langle e, x\rangle) \cong \varphi_e(x) \cong F(\mu y.T(e, x, y))$$

- Only μ operator
- Function index
- Execution flow
- Input
- Checker

Canonical form and interpretation

Kleene's Normal Form

There exist an elementary function F and a recursive primitive predicate T such that:

$$\forall e, x, \varphi_u(\langle e, x \rangle) \cong \varphi_e(x) \cong F(\mu y . T(e, x, y))$$

- Only μ operator
- Function index
- Execution flow
- Input
- Checker

Canonical form and interpretation

Kleene's Normal Form

There exist an elementary function F and a recursive primitive predicate T such that:

$$\forall e, x, \varphi_u(\langle e, x\rangle) \cong \varphi_e(x) \cong F(\mu y.T(e, x, y))$$

- Only μ operator
- Function index
- Execution flow
- Input
- Checker

Canonical form and interpretation

Kleene's Normal Form

There exist an elementary function F and a recursive primitive predicate T such that:

$$\forall e, x,\ \varphi_u\left(\langle e, x\rangle\right) \cong \varphi_e(x) \cong F(\ \mu y.T(e, x, y))$$

- Only μ operator
- Function index
- Execution flow
- Input
- Checker

Canonical form and interpretation

Kleene's Normal Form

There exist an elementary function F and a recursive primitive predicate T such that:

$$\forall e, x,\ \varphi_u(\langle e, x\rangle) \cong \varphi_e(x) \cong F(\mu y.T(e, x, y))$$

- Only μ operator
- Function index
- Execution flow
- Input
- Checker

Halt

Execution flow

Finite?

Bounded?

s_n^m and Fixed Point

s_n^m

There exists a recursive function s_n^m such that $\forall m, n, e$,

$$\varphi_e(\langle x_1, \ldots, x_n, y_1, \ldots, y_m \rangle) \simeq \varphi_{s_n^m(e, x_1, \ldots, x_n)}(\langle y_1, \ldots, y_m \rangle)$$

s_n^m and Fixed Point

s_n^m

There exists a recursive function s_n^m such that $\forall m, n, e$,

$$\varphi_e(\langle x_1, \ldots, x_n, y_1, \ldots, y_m \rangle) \simeq \varphi_{s_n^m(e, x_1, \ldots, x_n)}(\langle y_1, \ldots, y_m \rangle)$$

Fixed point

For each total recursive function *f* we can recursively compute an *n* such that:

$$\forall x, \varphi_n(x) \simeq \varphi_{f(n)}$$

s_n^m and Fixed Point

s_n^m

There exists a recursive function s_n^m such that $\forall m, n, e,$

Example

Let f be a function such that $\varphi_{f(n)} : x \mapsto \varphi_n(x) + 1$. Then there is an n verifying:

$$\forall x, \varphi_n(x) \cong \varphi_{f(n)}(x) \cong \varphi_n(x) + 1$$

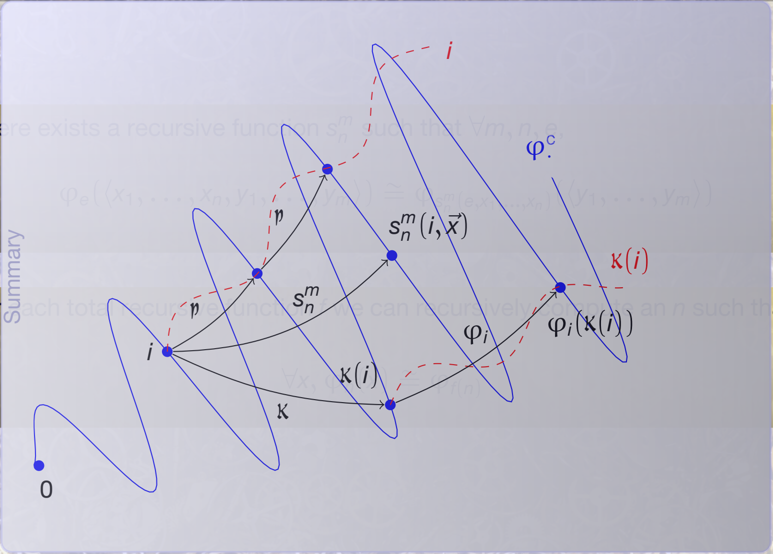
which is an index for the nowhere defined function.

Fixed point

For each total recursive function f we can recursively compute an n such that:

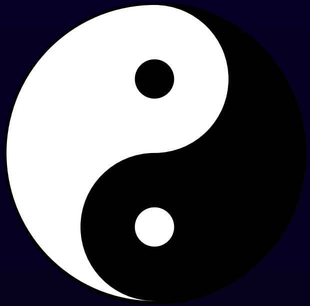
$$\forall x, \varphi_n(x) = \varphi_{f(n)}$$

s_n^m and Fixed Point



2. from total functions to partial computabilities

An interesting trade-off



Total function classes and recursive schemata

Closed classes of total functions

Class of total functions $\mathcal{C}$

Closed classes of total functions

Class of total functions $\mathcal{C}$

Constant functions

$\forall n, \forall x, \mathfrak{c}_n(x) = n$

Closed classes of total functions

Class of total functions $\mathfrak{C}$

Constant functions

$\forall n, \forall x, \mathfrak{c}_n(x) = n$

Projection and pairing functions

$\langle \cdot, \cdot \rangle, \pi^2_1, \pi^2_2 \in \mathfrak{C}$,
$\forall n_1, n_2, \pi^2_i(\langle n_1, n_2 \rangle) = n_i$

Closed classes of total functions

Class of total functions $\mathfrak{c}$

Constant functions

$\forall n, \forall x, \mathfrak{c}_n(x) = n$

Projection and pairing functions

$\langle \cdot, \cdot \rangle, \pi^2_1, \pi^2_2 \in \mathfrak{c}$,
$\forall n_1, n_2, \pi^2_i(\langle n_1, n_2 \rangle) = n_i$

Conditional operator

cond

Closed classes of total functions

Class of total functions $\mathbb{c}$

Constant functions

$\forall n, \forall x, \mathfrak{c}_n(x) = n$

Projection and pairing functions

$\langle \cdot, \cdot \rangle, \pi_1^2, \pi_2^2 \in \mathbb{c}$,
$\forall n_1, n_2, \pi_i^2(\langle n_1, n_2 \rangle) = n_i$

Conditional operator

cond

Stable under composition

$\forall f, g \in \mathbb{c}, f \circ g \in \mathbb{c}$.

Recursion schemata

Definition

Primitive recursive class: p

Smallest closed class stable by primitive recursion: $\mathbf{rec}_p$

Recursion schemata

Definition

Prim

Smal

itive r

Primitive recursion

For $g, h \in \mathrm{p}$,

$$f : n, \vec{x} \mapsto \mathbf{rec}_{\mathrm{p}}(g, h, n, \vec{x}) \in \mathrm{p},$$

with $\mathbf{rec}_{\mathrm{p}}(g, h, n, \vec{x})$ such that:

$$f(n, \vec{x}) = \begin{cases} h(\vec{x}) & \text{if } n = 0, \\ g(n, \vec{x}, f(n-1, \vec{x})) & \text{otherwise.} \end{cases}$$

Recursion schemata

Primitive recursion

$4 \quad f(4) = g(4, \vec{x}, f(4-1))$

$3 \quad f(3) = g(3, \vec{x}, f(3-1))$

$2 \quad f(2) = g(2, \vec{x}, f(2-1))$

$1 \quad f(1) = g(1, \vec{x}, f(1-1))$

$0 \quad f(0) = h(\vec{x})$

Recursion schemata

Definition

Primitive recursive class: p
Smallest closed class stable by primitive recursion: $\mathbf{rec}_{\mathsf{p}}$

Definition

α-recursive classes: c_{α}
Smallest closed class stable by α recursion: $\mathbf{rec}_{\alpha}$

Recursion schemata

Definition

Primitive recursive class: p

Smallest closed class stable by primitive recursion: $\mathbf{rec}_p$

Definition

α-recursive classes: c

Smallest closed class stable by α recursion: $\mathbf{rec}_\alpha$

α-recursion

For $g, h \in \mathrm{c}_\alpha$,

$$f : n, \vec{x} \mapsto \mathbf{rec}_{\alpha,\triangleleft}(g, h, n, \vec{x}) \in \mathrm{c}_\alpha$$

with $\mathbf{rec}_{\alpha,\triangleleft}(g, h, n, \vec{x})$ such that:

$$f(n, \vec{x}) = \begin{cases} g(n, \vec{x}, f(\theta(n, \vec{x}), \vec{x})) & \text{if } \overline{0} \triangleleft n \text{ and } \theta(n, \vec{x}) \triangleleft n, \\ h(n, \vec{x}) & \text{otherwise,} \end{cases}$$

where $\overline{\beta}$ stands for the ordinal notation (in $\triangleleft$) for β.

Recursion schemata

Definition **Prim**
Smal
itive

Definition **α-re**
Smal
cursi

α-recursion

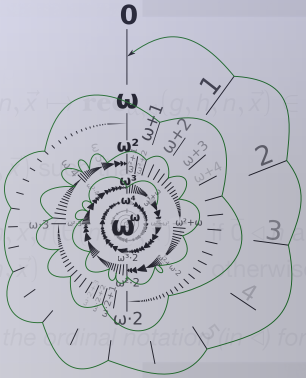
For $g, h \in c_\alpha$,

$$f : n, \vec{x} \mapsto \mathbf{rec}_{\alpha,\lhd}(g, h, n, \vec{x}) \in c_\alpha$$

with $\mathbf{rec}_{\alpha,\lhd}(g, h, n, \vec{x})$ such that

$$f(n, \vec{x}) = \begin{cases} g(n, \vec{x}, f(\theta(n, \vec{x}), \vec{x})) \text{ [illegible] } & \text{if } 0 \lhd \theta \text{ and } \theta(n, \vec{x}) \lhd n, \\ h(n, \vec{x}) & \text{otherwise,} \end{cases}$$

where $\overline{\beta}$ stands for the ordinal notation (in $\lhd$) for β.



Left behind

Loses

- Universal function not in c
- No unbounded search

Limits

- Limited function growth (e.g. Ackermann function not in p)
- Implies limited power

Can we get better results?

Primitive recursive coding schema

Indices

Toolbox for indices:

Compute new indices for a function

Compute index of composition

Primitive recursive coding schema

Toolbox for indices:

Indices

Compute new indices for a function

Compute index of composition

Requirements:

padding function: $\mathfrak{p}$

$\forall e, (\mathfrak{p}(e) > e) \wedge (\phi^{C}_{\mathfrak{p}(e)} = \phi^{C}_{e})$

composition function: $\mathfrak{c}$

$\forall e_1, e_2, \phi^{C}_{\mathfrak{c}(e_1,e_2)} = \phi^{C}_{e_1} \circ \phi^{C}_{e_2}$

Def: Coding schema

Primitive recursive coding schema

Indices

Toolbox for indices:

Compute new indices for a function

Compute index of composition

Simulation

Toolbox for simulation:

Check the validity of a flow

Bound the flow of a given function

Def: Coding schema

Requirements:

padding function: $\mathfrak{p}$

$$\forall e, (\mathfrak{p}(e) > e) \wedge (\phi^{c}_{\mathfrak{p}(e)} = \phi^{c}_{e})$$

composition function: $\mathfrak{c}$

$$\forall e_1, e_2, \phi^{c}_{\mathfrak{c}(e_1,e_2)} = \phi^{c}_{e_1} \circ \phi^{c}_{e_2}$$

Primitive recursive coding schema

Toolbox for indices:

Indices

Compute new indices for a function

Compute index of composition

Toolbox for simulation:

Simulation

Check the validity of a flow

Bound the flow of a given function

Requirements:

padding function: $\mathfrak{p}$

$\forall e, (\mathfrak{p}(e) > e) \wedge (\phi^{c}_{\mathfrak{p}(e)} = \phi^{c}_{e})$

composition function: $\mathfrak{c}$

$\forall e_1, e_2, \phi^{c}_{\mathfrak{c}(e_1,e_2)} = \phi^{c}_{e_1} \circ \phi^{c}_{e_2}$

step-by-step simulation: sim_c

$\forall x, e, \exists n, \phi^{c}_{e}(x) = \mathrm{sim}_c(e, x, n)$

cost function: use_c

$\forall x, e, \phi^{c}_{e}(x) = \mathrm{sim}_c(e, x, \phi^{c}_{\mathrm{use}_c(e)}(x))$

Def: Coding schema

Fundamental classes

A restriction of closed classes suiting our needs

Def: Fundamental class

Basis functions

Contains all the primitive recursive functions

Coding functions

Primitive recursive coding schema

Pairing functions

Projection functions

Closure

Stable by composition

Stable by primitive recursion

Enumeration

Tied to an enumeration $\phi^{C}_{.}$ (recursive) which is not in the class

How far did we get?

Thm

s^m_n for fundamental classes

Note: Primitive recursion is needed in order to obtain an homogeneous s.

How far did we get?

Thm **s_n^m for fundamental classes**

Note: Primitive recursion is needed in order to obtain an homogeneous s.

Thm **Still no recursion theorem**

It is actually impossible.

How far did we get?

Kleene and total functions

Recall our previous example:

Let f be a function such that $\varphi_{f(n)} : x \mapsto \varphi_n(x) + 1$. Then there is an n verifying:

$$\forall x, \varphi_n(x) \simeq \varphi_{f(n)}(x) \simeq \varphi_n(x) + 1$$

which is an index of the nowhere defined function.

Such a function cannot be total.

How far did we get?

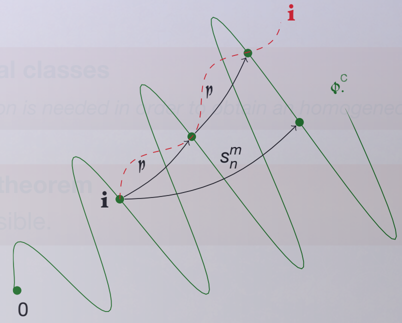
Thm s_n^m for fundamental classes

Note: Primitive recursion is needed in order to obtain an homogeneous s.

Thm **Still no recursion theorem**

It is actually impossible.

Summary



Rogers' Isomorphism Theorem

Ensures that our results do not depend on our choice of an enumeration

Rogers' Isomorphism Theorem

Ensures that our results do not depend on our choice of an enumeration

Go from an acceptable enumeration to another

Rogers' Isomorphism Theorem

Ensures that our results do not depend on our choice of an enumeration

Go from an acceptable enumeration to another

Thm

Myhill's isomorphism for fundamental classes

For A and B two sets of integers, f 1-1 from A to B and g 1-1 from B to A, we can build h an isomorphism between A and B.

Rogers' Isomorphism Theorem

Ensures that our results do not depend on our choice of an enumeration

Go from an acceptable enumeration to another

Thm **Myhill's isomorphism for fundamental classes**

For A and B two sets of integers, f 1-1 from A to B and g 1-1 from B to A, we can build h an isomorphism between A and B.

Thm **Rogers' isomorphism for fundamental classes**

Any acceptable enumeration is isomorphic to the canonical one.

Simulation, halt and domination

Primitive recursive case

Primitive recursive case

Ackermann function

Definition:

$$A : m, n \mapsto \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m, n-1)) & \text{otherwise} \end{cases}$$

Unary version:

$$\text{Ack} : n \mapsto A(n, n)$$

Simulation, halt and domination

Primitive recursive case

Ackermann function properties

Grows faster than any primitive recursive function

Simulation, halt and domination

Primitive recursive case

Ackermann function properties

Grows faster than any primitive recursive function

Enables us to bound the size of their flow

Simulation, halt and domination

Primitive recursive case

Ackermann function properties

Grows faster than any primitive recursive function

Enables us to bound the size of their flow

Universal simulation with sim_p and use_p:

$$x \longmapsto \mathrm{sim}_p(e, x, \mathrm{Ack}(f(\mathrm{use}_p(e), x)))$$

for some primitive recursive f

Simulation, halt and domination

General case for fundamental classes

Primitive recursive case

Ackermann function properties

Grows faster than any primitive recursive function

Enables us to bound the size of their flow

Universal simulation with sim_p and use_p:

$$x \mapsto \mathrm{sim}_p(e, x, \mathrm{Ack}(f(\mathrm{use}_p(e), x)))$$

for some primitive recursive f

Simulation, halt and domination

General case for fundamental classes

Busy Beaver function

Definition:

For an enumerable class C with $\mathrm{sim}_{\mathrm{C}}$ and $\mathrm{use}_{\mathrm{C}}$ functions:

$$\mathrm{BB}^{\phi}_{\mathrm{C}} = x \mapsto max \left\{ \phi^{\mathrm{C}}_{\mathrm{use}_{\mathrm{C}}(i)}(0) : i \leqslant x \right\} + x$$

Increasing version

Primitive recursive case

Ackermann function properties

Grows faster than any primitive recursive function

Enables us to bound the size of their flow

Universal simulation with $\mathrm{sim}_{\mathrm{p}}$ and $\mathrm{use}_{\mathrm{p}}$:

$$x \mapsto \mathrm{sim}_{\mathrm{p}}(e, x, \mathrm{Ack}(f(\mathrm{use}_{\mathrm{p}}(e), x)))$$

for some primitive recursive f

Simulation, halt and domination

General case for fundamental classes

Busy Beaver properties

Thm

Grows faster than any c-fundamental function

Enables us to bound the size of their flow

Universal simulation with sim_c and use_c:

$$x \longmapsto \text{sim}_c(e, x, \text{BB}_c^{\phi}(s_1^1(e, x)))$$

Primitive recursive case

Ackermann function properties

Grows faster than any primitive recursive function

Enables us to bound the size of their flow

Universal simulation with sim_p and use_p:

$$x \longmapsto \text{sim}_p(e, x, \text{Ack}(f(\text{use}_p(e), x)))$$

for some primitive recursive f

A recursive jump for fundamental classes

$\mathsf{BB}^{\phi}_{\mathsf{C}}$ allows us to totally compute any function in C.

Similar to the classical halting problem.

A recursive jump for fundamental classes

$\mathrm{BB}_{\mathrm{c}}^{\phi}$ allows us to totally compute any function in c.
Similar to the classical halting problem.

Def: Jump

Jump of c: $© = \mathrm{c}[\mathrm{BB}_{\mathrm{c}}^{\phi}]$

Still a fundamental class

$\phi_{\cdot}^{©}$ has a universal function for c.

A recursive jump for fundamental classes

$\text{BB}_{\mathrm{c}}^{\phi}$ allows us to totally compute any function in c.
Similar to the classical halting problem.

Def: Jump

Jump of c: $© = \mathrm{c}[\text{BB}_{\mathrm{c}}^{\phi}]$

Still a fundamental class

$\phi_{\cdot}^{©}$ has a universal function for c.

Thm

Kleene unbalanced theorem for a fundamental class c and any fundamental function $f \in \mathrm{c}$:

$$\exists n, \phi_n^{©} \simeq \phi_{f(n)}^{\mathrm{c}}$$

A recursive jump for fundamental classes

$\mathrm{BB}_{\mathrm{c}}^{\phi}$ allows us to totally compute any function in c.
Similar to the classical halting problem.

Def: Jump

Jump of c: $© = \mathrm{c}[\mathrm{BB}_{\mathrm{c}}^{\phi}]$

Still a fundamental class

$\phi_{\cdot}^{©}$ has a universal function for c.

Thm

Kleene unbalanced theorem for a fundamental class c and any fundamental function $f \in \mathrm{c}$:

$$\exists n, \phi_n^{©} \simeq \phi_{f(n)}^{\mathrm{c}}$$

Non-trivial sets and Rice's theorem

c-enumerability and c-recursivities

Enumerability and repetitions

Aim

Capture a class complexity through its enumerable sets

Aim

Capture a class complexity through its enumerable sets

Enumerable sets and p

Every enumerable set is enumerable by a primitive recursive function.

Enumerate $\mathbb{W}_e$

Simulate φ_e using bounded μ in Kleene's Normal Form:

$$\varphi_{e,s}(x) \simeq F(\mu y \leqslant s. T(e,x,y)) \simeq \mathrm{sim}_T(e,x,s)$$

Enumeration with repetitions:

$$\varphi_e(0), \varphi_e(0), \ldots, \varphi_e(0), \varphi_e(1), \varphi_e(2), \varphi_e(2), \ldots$$

Enumerability and repetitions

Aim

Capture a class complexity through its enumerable sets

Anoyance

Classical definition not interesting

Enumerable sets are c-enumerable.

Enumerability and repetitions

Aim

Capture a class complexity through its enumerable sets

Anoyance

Classical definition not interesting

Enumerable sets are c-enumerable.

Classical characterisations

Domain of a partial function

Range of a partial function

Enumerability and repetitions

Aim: Capture a class complexity through its enumerable sets

Anoyance: Classical definition not interesting

Enumerable sets are c-enumerable.

Classical characterisations

Domain of a partial function

Range of a partial function

Range of a total function

Enumerability and repetitions

Aim: Capture a class complexity through its enumerable sets

Anoyance: Classical definition not interesting

Enumerable sets are c-enumerable.

Classical characterisations

Domain of a partial function

Range of a partial function

Range of a total function

Range of a 1-1 function
(partial for finite sets)

Enumerability and repetitions

Aim

Capture a class complexity through its enumerable sets

Anoyance

Classical definition not interesting

Enumerable sets are c-enumerable.

Classical characterisations

Domain of a partial function

Range of a partial function

Range of a total function

Range of a 1-1 function
(partial for finite sets)

Produce a new element on each iteration

Enumerability and repetitions

Aim

Capture a class complexity through its enumerable sets

Anoyance

Classical definition not interesting

Enumerable sets are $\mathbb{c}$-enumerable.

Solution

A set is $\mathbb{c}$-enumerable if:

it is finite

or it is the range of a 1-1 $f \in \mathbb{c}$.

Enumeration: $\left(\mathrm{w}_e^{\mathbb{c}}\right)_{e\in\omega}$

Classical characterisations

Domain of a partial function

Range of a partial function

Range of a total function

Range of a 1-1 function
(partial for finite sets)

Produce a new element on each iteration

Enumerability and repetitions

Aim

Capture a class complexity through its enumerable sets

Anoyance

Classical definition not interesting

Enumerable sets are c-enumerable.

Solution

A set is c-enumerable if:

it is finite

or it is the range of a 1-1 $f \in \mathrm{c}$.

Enumeration: $(\mathrm{w}_e^{\mathrm{c}})_{e \in \omega}$

Tips & Tricks

How do we know if a function is 1-1?

We do not.

Check 1-1-ness for each new value

If not, the 1-1 prefix defines a finite set

Classical characterisations

Domain of a partial function

Range of a partial function

Range of a total function

Range of a 1-1 function *(partial for finite sets)*

Produce a new element on each iteration

c-recursivities

Aim

Extend the notion of c-enumerability to a notion of c-recursivity.

c-recursivities

Aim

Extend the notion of c-enumerability to a notion of c-recursivity.

Classical characterisations

A set E is recursive if:

- Its characteristic function χ_E is recursive
- E and $\overline{E}$ are enumerable
- E can be enumerated increasingly

c-recursivities

Aim Extend the notion of c-enumerability to a notion of c-recursivity.

Classical characterisations

A set E is recursive if:

Its characteristic function χ_E is recursive

E and $\overline{E}$ are enumerable

E can be enumerated increasingly

Def **χ-c-recursivity**

c-fundamental characteristic function

Def **weak-c-recursivity**

c-enumerable and co-c-enumerable

Def **strong-c-enumerability**

c-enumerable increasingly

Def **strong-c-recursivity**

Strongly c-enumerable and co-strongly c-enumerable

$\mathfrak{c}$-recursivities

Aim: Extend the notion of $\mathfrak{c}$-enumerability to a notion of $\mathfrak{c}$-recursivity.

Classical characterisations

A set E is recursive if:

Its characteristic function χ_E is recursive

E and $\overline{E}$ are enumerable

E can be enumerated increasingly

Def — **χ-$\mathfrak{c}$-recursivity**

$\mathfrak{c}$-fundamental characteristic function

Def — **weak-$\mathfrak{c}$-recursivity**

$\mathfrak{c}$-enumerable and co-$\mathfrak{c}$-enumerable

Def — **strong-$\mathfrak{c}$-enumerability**

$\mathfrak{c}$-enumerable increasingly

Def — **strong-$\mathfrak{c}$-recursivity**

Strongly $\mathfrak{c}$-enumerable and co-strongly $\mathfrak{c}$-enumerable

Thm — These notions are all different, and all compatible with the classical one

Noticeable sets and recursive properties

Def

Diagonal set

$\kappa_c^{\phi} = \{e : \phi_e^c(e) > 0\}$

Noticeable sets and recursive properties

Def — Diagonal set

$\kappa_c^\phi = \{e : \phi_e^c(e) > 0\}$

Def — Unary Ackermann range

$A = \mathrm{range}(\mathrm{Ack})$

Def — Busy Beaver range

$B_p = \mathrm{range}(\mathrm{BB}_p^\phi)$

$B_c = \mathrm{range}(\mathrm{BB}_c^\phi)$

Noticeable sets and recursive properties

Def — Diagonal set

$\kappa_c^\phi = \{e : \phi_e^c(e) > 0\}$

Def — Unary Ackermann range

$A = \text{range}(\text{Ack})$

Def — Busy Beaver range

$B_p = \text{range}(\text{BB}_p^\phi)$

$B_c = \text{range}(\text{BB}_c^\phi)$

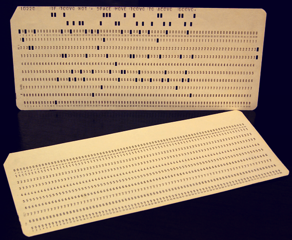
Thm

	r.e.	co-*r.e.*	p-e	s-p-e	co-p-e	co-s-p-e	w-p-rec	s-p-rec	χ-p-rec
κ	✓	✗	✓	✗	✗	✗	✗	✗	✗
κ_p^ϕ	✓	✓	✓	✗	✓	✗	✓	✗	✗
A	✓	✓	✗	✗	✓	✓	✗	✗	✓
B_p	✓	✓	✗	✗	✓	✓	✗	✗	✓

	r.e.	co-*r.e.*	c-e	s-c-e	co-c-e	co-s-c-e	w-c-rec	s-c-rec	χ-c-rec
κ	✓	✗	✓	✗	✗	✗	✗	✗	✗
κ_c^ϕ	✓	✓	✓	✗	✓	✗	✓	✗	✗
B_c	✓	✓	✗	✗	✓	✓	✗	✗	✓

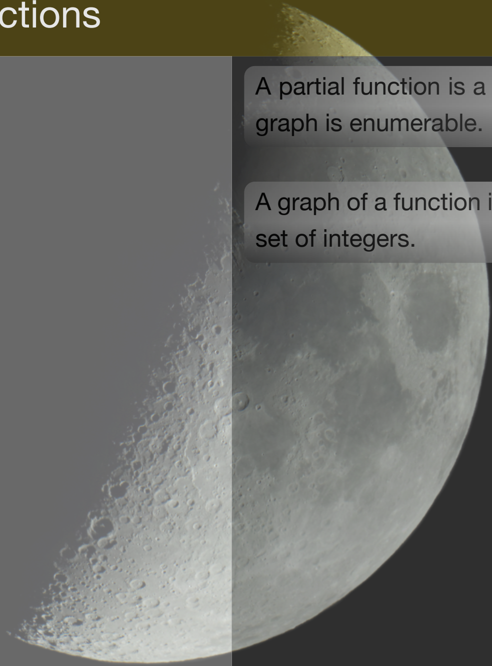
3. subcomputabilities

Computability, with holes



c-partial functions

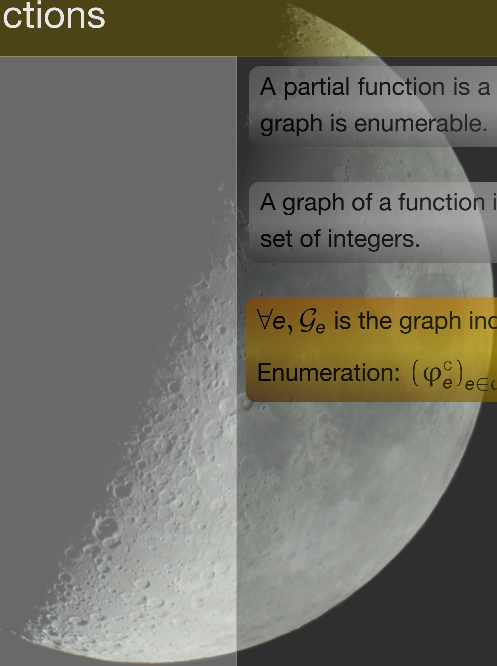
C-partial functions



A partial function is a function whose graph is enumerable.

A graph of a function is a well-formed set of integers.

c-partial functions



A partial function is a function whose graph is enumerable.

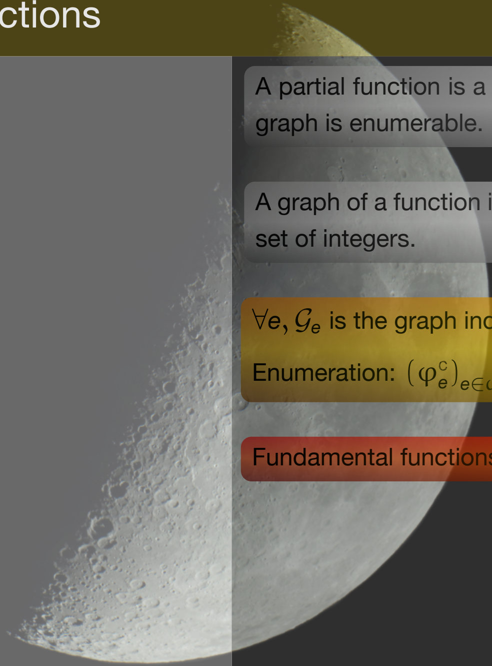
A graph of a function is a well-formed set of integers.

Def

$\forall e, \mathcal{G}_e$ is the graph induced by $\mathrm{w}_e^{\mathrm{c}}$

Enumeration: $(\varphi_e^{\mathrm{c}})_{e\in\omega}$

c-partial functions



A partial function is a function whose graph is enumerable.

A graph of a function is a well-formed set of integers.

Def

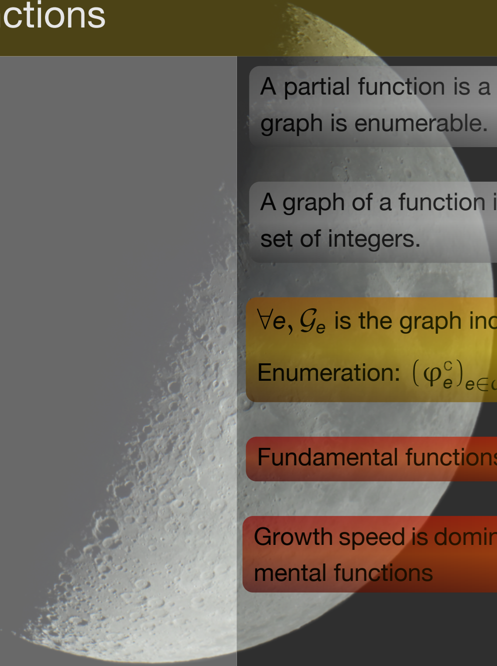
$\forall e, \mathcal{G}_e$ is the graph induced by w_e^{c}

Enumeration: $(\varphi_e^{c})_{e\in\omega}$

Thm

Fundamental functions are c-partials

c-partial functions



A partial function is a function whose graph is enumerable.

A graph of a function is a well-formed set of integers.

Def

$\forall e, \mathcal{G}_e$ is the graph induced by w_e^{c}

Enumeration: $\left(\varphi_e^{c}\right)_{e\in\omega}$

Thm

Fundamental functions are c-partials

Thm

Growth speed is dominated by fundamental functions

c-partial functions

Thm

Unusual closure

Not stable by composition

No s_n^m theorem

A partial function is a function whose graph is enumerable.

A graph of a function is a well-formed set of integers.

Def

$\forall e$, $\mathcal{G}_e$ is the graph induced by $\mathrm{w}_e^{\mathrm{c}}$

Enumeration: $(\varphi_e^{\mathrm{c}})_{e \in \omega}$

Thm

Fundamental functions are c-partials

Thm

Growth speed is dominated by fundamental functions

$\mathbb{C}$-partial functions

Unusual closure (Thm)

Not stable by composition

No s_n^m theorem

No composition (Counter examples)

$\mathbb{C}$ contains

$$f : n \mapsto \begin{cases} \mathrm{BB}_{\mathbb{C}}^{\phi}(p) & \text{if } n = 2p \\ p & \text{if } n = 2p + 1 \end{cases}$$

and $g : n \mapsto 2n$, but not $f \circ g = \mathrm{BB}_{\mathbb{C}}^{\phi}$.

No s_n^m theorem

$\mathbb{C}$ contains $f : \langle e, x \rangle \mapsto \varphi_e(x)$ but not all the recursive functions.

A partial function is a function whose graph is enumerable.

A graph of a function is a well-formed set of integers.

(Def) $\forall e$, G_e is the graph induced by φ[illegible]

Enumeration: $(\varphi_e^{\mathbb{C}})_{e \in \omega}$

(Thm) ...als

(Thm) Growth speed is dominated by fundamental functions

c-partial functions

Thm

Unusual closure

Not stable by composition

No s_n^m theorem

Thm

Non-trivial c-creativity/productivity notion

A partial function is a function whose graph is enumerable.

A graph of a function is a well-formed set of integers.

Def

$\forall e, \mathcal{G}_e$ is the graph induced by $\mathrm{w}_e^{\mathrm{c}}$

Enumeration: $(\varphi_e^{\mathrm{c}})_{e\in\omega}$

Thm

Fundamental functions are c-partials

Thm

Growth speed is dominated by fundamental functions

c-partial functions

Thm

Unusual closure

Not stable by composition

No s_n^m theorem

Thm

Non-trivial c-creativity/productivity notion

Thm

Partial Kleene's second recursion theorem for a fundamental class c

For $f \in$ c and h c-partial of domain A co-enumerable,

$\exists n$ s.t. $(\varphi_n^{c}) \restriction_{\overline{A}} \simeq \left(\varphi_{f(n)}^{c}\right) \restriction_{\overline{A}}$ and $(\varphi_n^{c}) \restriction_{A} \simeq h$

A partial function is a function whose graph is enumerable.

A graph of a function is a well-formed set of integers.

Def

$\forall e, \mathcal{G}_e$ is the graph induced by w_e^{c}

Enumeration: $(\varphi_e^{c})_{e \in \omega}$

Thm

Fundamental functions are c-partials

Thm

Growth speed is dominated by fundamental functions

c-partial functions

Proof

Let $f = \phi^{c}_{i_f} \in c$ and $h = \varphi^{c}_{a}$ c-partial of domain A co-enumerable.

Goal: show that there is a c-partial fixed-point with fundamental computable index.
The following function is c-partial as a recursive extention of a c-partial:

$$u \mapsto \psi_x(u) = \begin{cases} \varphi^{c}_{a}(u) & \text{if } u \in A \\ \varphi^{c}_{\varphi^{c}_{x}(x)}(u) & \text{otherwise.} \end{cases}$$

Its index is computable from x in c, by a function d_a of index i_{d_a}.
Let $e_a = \mathbf{comp}(i_f, i_{d_a})$ be an index for $f \circ d_a$.

$$\forall u, u \mapsto \varphi^{c}_{d_a(e_a)}(u) \simeq \begin{cases} \varphi^{c}_{a}(u) & \text{if } u \in A \\ \varphi^{c}_{\varphi^{c}_{e_a}(e_a)}(u) \simeq \varphi^{c}_{f \circ d_a(e_a)}(u) & \text{otherwise.} \end{cases}$$

Choose $n = d_a(e_a)$, then:
$\varphi^{c}_{n} \simeq \varphi^{c}_{f(n)}$ on $\overline{A}$ and $\varphi^{c}_{n} \simeq \varphi^{c}_{a} \simeq h$ on A.
Hence, n is a partial fixed-point for f.

Let $f = \varphi^c_{i_f} \in$ c and $h = \varphi^c_a$ c-partial of domain A co-enumerable.

Goal: show that there is a c-partial fixed-point with fundamental computable index.

The following function is c-partial as a recursive extention of a c-partial:

$$u \mapsto \psi_x(u) = \begin{cases} \varphi^c_a(u) & \text{if } u \in A \\ \varphi^c_{\varphi^c_x(x)}(u) & \text{otherwise.} \end{cases}$$

Its index is computable from x in c by a function d_a of index i_{d_a}.

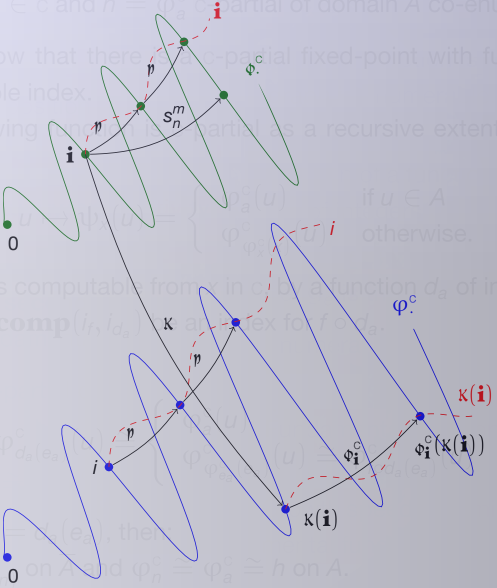
Let $e_a = \mathbf{comp}(i_f, i_{d_a})$ be an index for $f \circ d_a$.

$$\forall u, u \mapsto \varphi^c_{d_a(e_a)}(u) = \begin{cases} \varphi^c_a(u) & \text{if } u \in A \\ \varphi^c_{\varphi^c_{e_a}(e_a)}(u) = \varphi^c_{f(d_a(e_a))}(u) & \text{otherwise.} \end{cases}$$

Choose $n = d_a(e_a)$, then:

$\varphi^c_n \simeq \varphi^c_{f(n)}$ on $\bar{A}$ and $\varphi^c_n \simeq \varphi^c_a \simeq h$ on A.

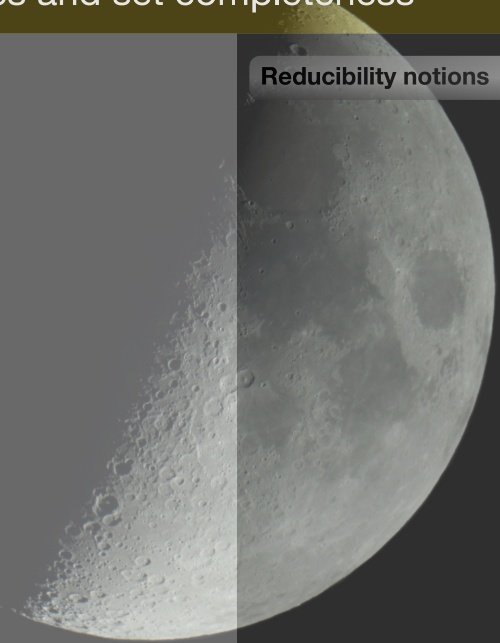
Hence, n is a partial fixed-point for f.



Summary

Reducibilities and set completeness

Reducibility notions



Reducibilities and set completeness

Def **χ-c-recursivity**

c-fundamental characteristic function

Def **weak-c-recursivity**

c-enumerable and co-c-enumerable

Def **strong-c-enumerability**

c-enumerable increasingly

Def **strong-c-recursivity**

Strongly c-enumerable and co-strongly c-enumerable

Reducibility notions

Reducibilities and set completeness

Def — χ-c-recursivity

c-fundamental characteristic function

Def — weak-c-recursivity

c-enumerable and co-c-enumerable

Def — strong-c-enumerability

c-enumerable increasingly

Def — strong-c-recursivity

Strongly c-enumerable and co-strongly c-enumerable

Reducibility notions

Def — $A \leqslant^{\chi}_{\text{c-T}} B$

If $\chi_A \in \mathrm{c}[\chi_B]$.

Def — $A \leqslant^{w}_{\text{c-T}} B$

If $\forall \mathfrak{e}_B, \mathfrak{e}_{\overline{B}}, \exists \mathfrak{e}_A, \mathfrak{e}_{\overline{A}} \in \mathrm{c}[\mathfrak{e}_B, \mathfrak{e}_{\overline{B}}]$.

Def — $A \leqslant^{s}_{\text{c-e}} B$

If $\exists \mathfrak{p}_A \in \mathrm{c}[\mathfrak{p}_B]$.

Def — $A \leqslant^{s}_{\text{c-T}} B$

If $\exists \mathfrak{p}_A, \mathfrak{p}_{\overline{A}} \in \mathrm{c}[\mathfrak{p}_B, \mathfrak{p}_{\overline{B}}]$.

Reducibilities and set completeness

Def

$$\kappa_{\mathrm{c}} = \{e : \varphi_e^{\mathrm{c}}(e) \downarrow\}$$

Thm κ_{c} is *many-one*-complete *via* c-partial reductions.

Reducibility notions

Def $A \leqslant^{\chi}_{\text{c-}\mathbf{T}} B$

If $\chi_A \in \mathrm{c}[\chi_B]$.

Def $A \leqslant^{w}_{\text{c-}\mathbf{T}} B$

If $\forall \mathfrak{e}_B, \mathfrak{e}_{\overline{B}}, \exists \mathfrak{e}_A, \mathfrak{e}_{\overline{A}} \in \mathrm{c}[\mathfrak{e}_B, \mathfrak{e}_{\overline{B}}]$.

Def $A \leqslant^{s}_{\text{c-}\mathbf{e}} B$

If $\exists \mathfrak{p}_A \in \mathrm{c}[\mathfrak{p}_B]$.

Def $A \leqslant^{s}_{\text{c-}\mathbf{T}} B$

If $\exists \mathfrak{p}_A, \mathfrak{p}_{\overline{A}} \in \mathrm{c}[\mathfrak{p}_B, \mathfrak{p}_{\overline{B}}]$.

Reducibilities and set completeness

Proof

$\kappa \leqslant_m \kappa_{\mathrm{c}}$ via a c-fundamental reduction

For a and z such that φ^{c}_a and φ^{c}_z are resp. never or always null, and $f_x \in \mathrm{c}$ such that $\forall e$:

$$\varphi^{\mathrm{c}}_{f_x(e)} : y \mapsto \begin{cases} \varphi_x(x) & \text{if } y = a \text{ or } y = e \\ 0 & \text{otherwise.} \end{cases}$$

Let A be the strongly-c-enumerable set $\{\mathfrak{p}^{2n}(z) : n > 0\}$, and h a function null on A and undefined on $\overline{A}$.

By our partial Kleene, we have:

$$\varphi^{\mathrm{c}}_n : y \mapsto \begin{cases} 0 & \text{if } y \in A \\ \varphi^{\mathrm{c}}_{f_x(n)}(y) & \text{otherwise.} \end{cases}$$

By case analysis we can verify that $n \in \kappa_{\mathrm{c}} \leftrightarrow x \in \kappa$, with n being c-fundamentally computable from x.

4. towards a fine structure of computabilities

Rising above

To infinity and beyond

Hyperstructure of fundamental classes

Relativisations of Kleene's $\mathcal{O}$ and Hyperarithmetic sets

A notion of $\mathfrak{c}$-recursive orders (ordinals)

A fine hierarchy of $\mathfrak{c}$-degrees

Conjecture

A bottom-up (complexity-wise) construction of enumerable degrees

Fragments above computability

The case of Σ-recursion

Functions over sets in admissible levels of Gödel's L hierarchy

An enumeration $\left(\phi_e^{\mathbb{A}}\right)_{e\in\alpha}$ of Δ_0 (α-finite) sets

Plays the role of fundamental functions

An enumeration $\left(\varphi_e^{\mathbb{A}}\right)_{e\in\alpha}$ of Σ_1 (α-enumerable) sets

Plays the role of partial functions

Preliminary results

s_n^m-like theorem

Fixed-point theorem

Perspectives and conclusion

A general computability framework

for studying subrecursion and beyond

Perspectives and conclusion

A general computability framework

for studying subrecursion and beyond

Applications

Relativised notion of Kolmogorov complexity

General fine structure

Study of the c-recursive ordinals

Ordinal iterations of the jump

Proof theory

Links between c-degrees and honest degrees

Yielding results about minimal independent statements using c classes?

Thank you for your attention.

Credits

Villeneuve De Berg: F. Givors
Gears: D. Proulx
Calanques: B. Monginoux
Autumn: F. Givors
Pic Saint Loup: Ophrys34
Moon: Thomas Bresson
Punched card: José Antonio González Nieto
Giraffe: Rob Hooft
Milky Way: NASA
Railway tracks: Arne Hückelheim