Infinite Time Turing Machines for elementary proofs on recursive reals

Journée des Arithmétiques Faibles

Kenza Benjelloun

Joint work with Bruno Durand

Université de Côte d'Azur

Università Degli Studi di Trieste

11 septembre 2024

Outline of talk

Preliminary notions

Infinite Time Turing machines and algorithmic tools

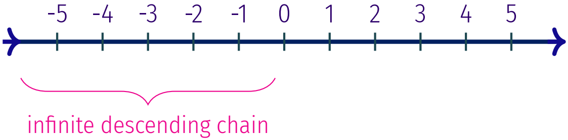
Our elementary proof of Harrison’s theorem

Prooving that our proof is elementary

Table of Contents

Order relations

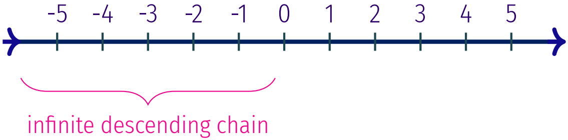
$\mathbb{Z}$:



For $a, b \in \mathbb{Z}$, a natural relation : $a <_{\mathbb{Z}} b$
We denote orders by $\prec$ here $a \prec b \iff a <_{\mathbb{Z}} b$

Order relations

$\mathbb{Z}$:



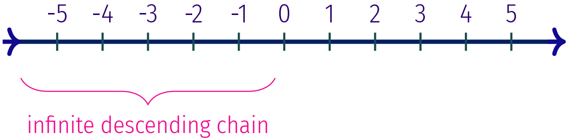
For $a, b \in \mathbb{Z}$, a natural relation : $a <_{\mathbb{Z}} b$
We denote orders by $\prec$ here $a \prec b \iff a <_{\mathbb{Z}} b$

This is an order (strict)

- *Anti-reflexive* ($x \not\prec x$)
- *Linear* (any two elements are comparable)
- *Transitive* ($\forall abc\ a \prec b \wedge b \prec c \Rightarrow a \prec c$)

Order relations

$\mathbb{Z}$:



For $a, b \in \mathbb{Z}$, a natural relation : $a <_{\mathbb{Z}} b$
We denote orders by $\prec$ here $a \prec b \iff a <_{\mathbb{Z}} b$

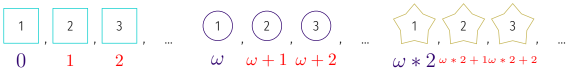
This is an order (strict)

- ∘ *Anti-reflexive* ($x \not\prec x$)
- ∘ *Linear* (any two elements are comparable)
- ∘ *Transitive* ($\forall abc\ a \prec b \wedge b \prec c \Rightarrow a \prec c$)
- × **Not** a *well-order* – $\mathbb{Z}$ has no least element

- well-orders : any non-empty subset has a least element
- well-order $\iff$ no infinite descending chain (Zorn - Choice)

Orders and real codes

- Ordering *countable* sequences of elements...

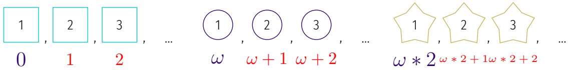


$$1-\frac{1}{2^1} < 1-\frac{1}{2^2} < 1-\frac{1}{2^3} < \dots \quad 2-\frac{1}{2^1} < 2-\frac{1}{2^2} < 2-\frac{1}{2^3} < \dots \quad 3-\frac{1}{2^1} < 3-\frac{1}{2^2} < 3-\frac{1}{2^3} < \dots$$

$$0 \quad 1 \quad 2 \quad \omega \quad \omega+1 \quad \omega+2 \quad \omega*2 \quad \omega*2+1 \quad \omega*2+2$$

Orders and real codes

- Ordering *countable* sequences of elements...



$$1-\frac{1}{2^1} < 1-\frac{1}{2^2} < 1-\frac{1}{2^3} < \dots \quad 2-\frac{1}{2^1} < 2-\frac{1}{2^2} < 2-\frac{1}{2^3} < \dots \quad 3-\frac{1}{2^1} < 3-\frac{1}{2^2} < 3-\frac{1}{2^3} < \dots$$

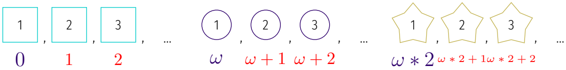
$$0 \quad 1 \quad 2 \quad \omega \quad \omega+1 \quad \omega+2 \quad \omega*2 \quad \omega*2+1 \quad \omega*2+2$$

- The objects we order are represented by integers
- Encoding pairs : $\langle a, b\rangle$ is an integer too. e.g. using *Cantor's pairing function* :

$$\langle a, b\rangle = \frac{(a+b)(a+b+1)}{2} + b$$

Orders and real codes

- Ordering *countable* sequences of elements...



$1-\frac{1}{2^1} <$	$1-\frac{1}{2^2} <$	$1-\frac{1}{2^3} < \dots$	$2-\frac{1}{2^1} <$	$2-\frac{1}{2^2} <$	$2-\frac{1}{2^3} < \dots$	$3-\frac{1}{2^1} <$	$3-\frac{1}{2^2} <$	$3-\frac{1}{2^3} < \dots$
0	1	2	ω	$\omega+1$	$\omega+2$	$\omega * 2$	$\omega * 2+1$	$\omega * 2+2$

- The objects we order are represented by integers
- Encoding pairs : $\langle a, b\rangle$ is an integer too. e.g. using *Cantor's pairing function* :

$$\langle a, b\rangle = \frac{(a+b)(a+b+1)}{2} + b$$

- We encode relations by reals

$$x = 0100010101010101...0 \iff (x_{\langle a,b\rangle} = 1 \Leftrightarrow a \prec b)$$

⟶ This real encodes a relation, an order, an ordinal

Ordinals encoded as reals

Let x be a real code,
• x is **total** if all pairs of integers $n = \langle a, b \rangle$ are comparable
• the **domain** of x is the set of integers appearing in at least one comparable pair $a \prec_x b$ of the order encoded by x
◦ well-orders are always comparable – an **ordinal** is a class of well orders of same length (length is called *ordinal type*).

Ordinals encoded as reals

Let x be a real code,

• x is **total** if all pairs of integers $n = \langle a, b \rangle$ are comparable

• the **domain** of x is the set of integers appearing in at least one comparable pair $a \prec_x b$ of the order encoded by x

◦ well-orders are always comparable – an **ordinal** is a class of well orders of same length (length is called *ordinal type*).

Proposition

We can recursively transform any (not-total) enumerable (infinite) code into a total enumerable code

Ordinals encoded as reals

Let x be a real code,

• x is **total** if all pairs of integers $n = \langle a, b \rangle$ are comparable

• the **domain** of x is the set of integers appearing in at least one comparable pair $a \prec_x b$ of the order encoded by x

∘ well-orders are always comparable – an **ordinal** is a class of well orders of same length (length is called *ordinal type*).

Proposition

We can recursively transform any (not-total) enumerable (infinite) code into a total enumerable code

• x is a **recursive real** if $n \mapsto x_n$ is computable by a Turing machine

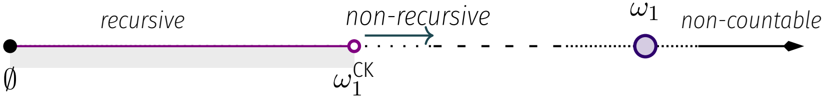
• x is an **enumerable real** if 1's in x are enumerable by a Turing machine

• α is a **recursive ordinal** if it has at least one recursive encoding

recursive ordinals

Recursive ordinals form an **initial segment** of countable ordinals :

$$\sup_{\alpha \in \mathsf{REC}} \alpha = \omega_1^{\mathsf{CK}}$$



Theorem (Spector Theorem, 1958)

The order type of a Σ_1^1 well-order is less than ω_1^{CK}

$\rightarrow$ **collapse theorem**

Collapse on recursive ordinals

Theorem (First order improved version)

If an ordinal α is arithmetic, then it is also recursive. Furthermore, the transformation of the formula that caracterizes α into a program (for $\omega \cdot \alpha$) is recursive.

$\rightarrow$ arithmetic ordinals are those ordinals encoded by a real defined by an arithmetic formula
$\rightarrow$ "α is also recursive" means that there exists another real that codes a well-order of same length α which is recursive

later we'll come back on the collapse and its role in our construction

Table of Contents

Infinite Time Turing Machines definition (ITTM)

- Developped by Joel David Hamkins and Andy Lewis in 2000
- Analog of a Turing machine but with ordinal time
- Tapes of size ω

Infinite Time Turing Machines definition (ITTM)

- Developped by Joel David Hamkins and Andy Lewis in 2000
- Analog of a Turing machine but with ordinal time
- Tapes of size ω

Ordinal time :

(1) Initial point : $\emptyset$
(2) Successor case : $\alpha \cup \{\alpha\}$
(3) Limit case : $\sup_{\beta<\alpha} = \bigcup \beta \in \alpha$

Infinite Time Turing Machines definition (ITTM)

- Developped by Joel David Hamkins and Andy Lewis in 2000
- Analog of a Turing machine but with ordinal time
- Tapes of size ω

Ordinal time :

(1) Initial point : $\emptyset$

(2) Successor case : $\alpha \cup \{\alpha\}$

(3) Limit case : $\sup_{\beta<\alpha} = \bigcup \beta \in \alpha$

ITTM computation steps :

(1) Starting time : 0, initial state q_0

(2) Successor case : normal Turing machine operation

(3) Limit case :

Infinite Time Turing Machines definition (ITTM)

- Developped by Joel David Hamkins and Andy Lewis in 2000
- Analog of a Turing machine but with ordinal time
- Tapes of size ω

Ordinal time :

(1) Initial point : $\emptyset$
(2) Successor case : $\alpha \cup \{\alpha\}$
(3) Limit case : $\sup_{\beta<\alpha} = \bigcup \beta \in \alpha$

ITTM computation steps :

(1) Starting time : 0, initial state q_0
(2) Successor case : normal Turing machine operation
(3) Limit case :
    - ⋆ `limsup` on all cells (alphabet $\{0, 1\}$)
    - ⋆ `rewind` the head at the begining of the tape
    - ⋆ `goto` limit state q_L

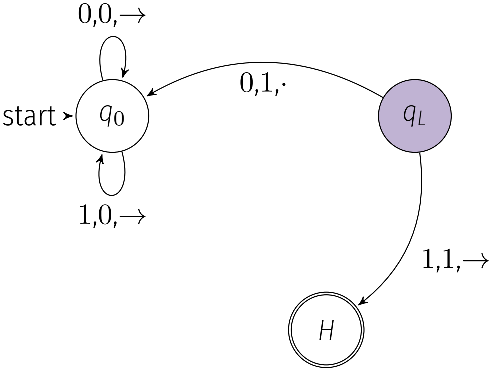


Figure – States of an ITTM

Flash algorithm on an ITTM

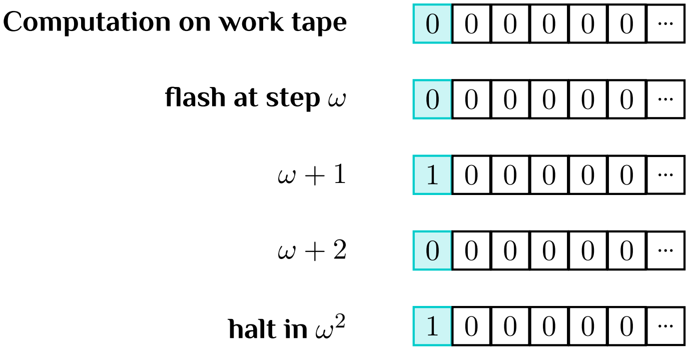


Figure – Evolution of the work tape of an ITTM

Flash algorithm on an ITTM

Computation on work tape	0	0	0	0	0	0	...
flash at step ω	0	0	0	0	0	0	...
$\omega + 1$	1	0	0	0	0	0	...
$\omega + 2$	0	0	0	0	0	0	...
halt in ω^2	1	0	0	0	0	0	...

Figure – Evolution of the work tape of an ITTM

- `input` : a real that we place on a tape
- `output` : a real that we read from a tape
- an ITTM computes a function from $\mathbb{R}$ to $\mathbb{R}$

Counting through orders

Definition (clockable ordinal)

α is a clockable ordinal if and only if $\exists$ an ITTM μ such that the computation of μ (on empty input $000\ldots$) halts in exactly α steps.

Beware : clockable ordinals do not form a segment. You can have $\alpha < \beta$ with β clockable and α not.

Theorem (Count-Through Theorem - Hamkins and Lewis, 2000)

There exists an ITTM that, given the real representation of a linear order, decides if the order is a well order in time $\alpha + \omega$ where α is the length of the w.o.i.s (well order initial segment).

Our construction

- We design an ITTM μ that takes as input a program of a Turing machine *n*
  - μ simulates the TM number *n* and checks whether it is the program for a recursive real *x*

    Turing machine simulation by ITTM - time ω
  - If yes, μ checks that *x* codes a linear order

    Simple algorithmics on ITTM, timers - time ω
  - If yes, μ counts through the order to check whether the order is a well order

    Count-Through Theorem - time $\alpha + \omega$

Our construction

- We design an ITTM μ that takes as input a program of a Turing machine *n*
  - μ simulates the TM number *n* and checks whether it is the program for a recursive real *x*

    Turing machine simulation by ITTM - time ω
  - If yes, μ checks that *x* codes a linear order

    Simple algorithmics on ITTM, timers - time ω
  - If yes, μ counts through the order to check whether the order is a well order

    Count-Through Theorem - time $\alpha + \omega$
- We ITTM-compute $\mu(0), \mu(1), \mu(2), \ldots$
- Let us suppose now that all such α are recursive. We conclude that our computation halts in *exactly* ω_1^{CK}
- ω_1^{CK} is *not* clockable $\Rightarrow$ contradiction !!

Our construction

- We design an ITTM μ that takes as input a program of a Turing machine *n*
  - μ simulates the TM number *n* and checks whether it is the program for a recursive real *x*

    Turing machine simulation by ITTM - time ω
  - If yes, μ checks that *x* codes a linear order

    Simple algorithmics on ITTM, timers - time ω
  - If yes, μ counts through the order to check whether the order is a well order

    Count-Through Theorem - time $\alpha + \omega$
- We ITTM-compute $\mu(0), \mu(1), \mu(2), \ldots$
- Let us suppose now that all such α are recursive. We conclude that our computation halts in *exactly* ω_1^{CK}
- ω_1^{CK} is *not* clockable $\Rightarrow$ contradiction !!

  beware of this high level argument
- We conclude that there exists a recursive linear order the *w.o.i.s* of which has ordinal type ω_1^{CK} and we prove **Harrison's theorem**

Definition (pseudo-well ordering)

A pseudo-well ordering is a linear order that has **no** *infinite* **arithmetic** *descending chain.*

- In other terms, one cannot arithmetically differentiate a pseudo-well ordering from a well-order
- Recursive pseudo-well ordering have a *w.o.i.s* of length ω_1^{CK} (Gandy - easier proof by ITTM)
- A recursive linear order the *w.o.i.s* of which has length ω_1^{CK} is a recursive pseudo-well ordering
- Harrison proved in 1968 that there exists recursive pseudo-well orderings
- We call "Harrison real" a recursive real encoding a pseudo-well ordering

Insight : Pseudo-well orderings and Harrison's reals challenge our intuitions about linear orders and the nature of recursive sets.

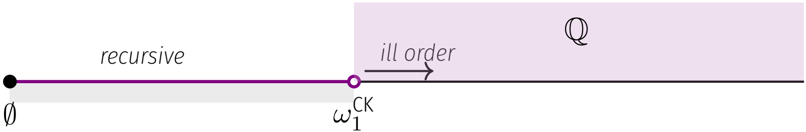


Figure – An example of a recursive pseudo-well ordering

Table of Contents

Some formulations of Harrison's theorem

Theorem (Original formulation)

There exists a recursive linear order of which the ordinal type of the w.o.i.s is exactly ω_1^{CK}.

Some formulations of Harrison's theorem

Theorem (Original formulation)

There exists a recursive linear order of which the ordinal type of the w.o.i.s is exactly ω_1^{CK}.

Theorem (First order formulation)

There exists a recursive real that codes a linear order so that any recursive ordinal is a prefix of this order.

Motivations

Original proof and classical proofs use descriptive set theory and computability results such as

$$\Sigma^1_1 \neq \Pi^1_1$$

they are rather simple but lays in the analytic hierarchy.

Some other second order proofs exist, for instance using Kleene-Brouwer order on a tree without arithmetic infinite path.

Our proof is bottom-up (”constructive”), more *elementary*.

$\longrightarrow$ We prove this now.

Table of Contents

Our goal is to place our proof in ACA_0

$\mathbf{ACA}_0$ stands for **Arithmetical Comprehension Axiom**

It allows comprehension (existence) of sets defined by arithmetical formulas

It contains basic Peano arithmetics RCA_0

More formally : in ACA_0, for all arithmetical formula $\varphi(n)$,
we have a set $A = \{n \mid \varphi(n)\}$

ACA_0 deals well with recursive sets and computations that involve recursive ordinals

The result $\Sigma_1^1 \neq \Pi_1^1$ states that there exists a Σ_1^1 set of reals that is not Π_1^1.

Σ_1^1 : Sets definable by an existential quantifier over all arithmetical predicates

Π_1^1 : Sets definable by a universal quantifier over all arithmetical predicates

This is a separation result in the **analytical hierarchy**

Why ACA_0 is Too Weak to Prove $\Sigma^1_1 \neq \Pi^1_1$

- ACA_0 only provides comprehension for **arithmetical formulas**
- Σ^1_1 and Π^1_1 sets require quantification over **all sets of natural numbers**, which is beyond the arithmetical realm
- the separation $\Sigma^1_1 \neq \Pi^1_1$ requires constructing a set that cannot be defined by any arithmetical formula
- ACA_0 lacks the necessary strength to express such higher-level definitions, rendering this theory too weak to contain the original proofs
- It seems that TRA is necessary, and so the proof by Harrison and others reside in ATR_0

our ITTM computations

- ◦ μ simulates the TM number *n* and checks whether it is the program for a recursive real *x*

  Turing machine simulation by ITTM - time ω
- ◦ If yes, μ checks that *x* codes a linear order

  Simple algorithmics on ITTM, timers - time ω
- ◦ If yes, μ counts through the order to check whether the order is a well order

  Count-Through Theorem - time $\alpha + \omega$

- As for Turing machines, there is an equivalence between ITTM computations and some class of formula
- If an ITTM halts in time α, then it computes a function in $\Sigma_1(L_\alpha)$. L_α is Gödel constructible set (Hamkins and Lewis 2000)

- ◦ We have used the fact that all ordinals below ω_1^{CK} are recursive – thus our ITTM enumerates them all, and make computation steps at each stage
- ◦ We need our first-order recursive version of the collapse theorem
- ◦ If an ordinal α is arithmetic, then it is also recursive. Furthermore, the transformation of the formula that caracterizes α into a program (for $\omega \cdot \alpha$) is recursive
- This result was already known in recursion theory, but we found no first order proof in litterature. We prove this using a subtle *priority argument* construction on orders.

- We remark that a Σ_1 sum of recursive ordinals is also recursive
- We use the fact that no ITTM may halt in ω_1^{CK}. The classical justification is that $\omega_1^{\mathsf{CK}}$is an *admissible* ordinals and that halting of an ITTM would contradict admissibility. This argument is too high !

◦ We replace the last argument by the combination of two facts :

ITTM computations in time β lays in $\Sigma_1(L_\beta)$
a Σ_1 sum of recursive ordinals is also recursive

$$\tau = \sum_{n \in \omega} \tau_n$$

ITTM algorithmic argument

$$\tau = \sum_{n \in \omega} \tau_n$$

- For all recursive ordinals β, there exists a TM b which produces the code x_β
- $\beta \leq \tau_b \leq \omega_1^{\mathrm{CK}} \Rightarrow \omega_1^{\mathrm{CK}} \leq \tau$
- $\forall n\ \tau_n$ is Σ_1-defined over L_{τ_n}

ITTM algorithmic argument

$$\tau = \sum_{n \in \omega} \tau_n$$

- For all recursive ordinals β, there exists a TM b which produces the code x_β
- $\beta \leq \tau_b \leq \omega_1^{\mathrm{CK}} \Rightarrow \omega_1^{\mathrm{CK}} \leq \tau$
- $\forall n\ \tau_n$ is Σ_1-defined over L_{τ_n}
- $\mu(n)$ computation is $\Sigma_1(L_{\tau_n}) \longrightarrow$ which is arithmetical
- By the collapse theorem, τ_n is recursive and $\tau = \omega_1^{\mathrm{CK}}$

ITTM algorithmic argument

$$\tau = \sum_{n \in \omega} \tau_n$$

- For all recursive ordinals β, there exists a TM b which produces the code x_β
- $\beta \leq \tau_b \leq \omega_1^{\mathrm{CK}} \ \Rightarrow \ \omega_1^{\mathrm{CK}} \leq \tau$
- $\forall n \ \tau_n$ is Σ_1-defined over L_{τ_n}
- $\mu(n)$ computation is $\Sigma_1(L_{\tau_n}) \longrightarrow$ which is arithmetical
- By the collapse theorem, τ_n is recursive and $\tau = \omega_1^{\mathrm{CK}}$

Quod erat demonstrandum

Our entire proof resides in ACA_0

Vielen Dank fr Ihre Aufmerkſamkeit

Merci pour votre attention

Thank you for liſtening