Complexity and Decision Times for ITTMs
The Story of the Bold Conjecture

(joint work with P. Schlicht and P. Welch)

Outline

Complexity and Decision Times for ITTMs - The Story of the “Bold Conjecture”
(joint work with Philipp Schlicht and Philip Welch)

Models of Transfinite Computability

Several machine models of transfinite computation generalize Turing machines to infinite time or space (*ITTMs*, α-TMs, *OTMs*...)

We also consider generalizations of a different model, the so-called unlimited register machines (URM's):

Definition:

An unlimited register machine (URM) has registers $R_0, R_1, \ldots$ which can hold natural numbers. A URM program is a finite list $P = I_0, I_1, ..., I_{s-1}$ of instructions, each of which may be of one of five kinds:

URM-programs

- the zero instruction $Z(n)$ changes the contents of R_n to 0, leaving all other registers unaltered;
- the successor instruction $S(n)$ increases the natural number contained in R_n by 1, leaving all other registers unaltered;
- the oracle instruction $O(n)$ replaces the content of the register R_n by the number 1 if the content is an element of the oracle, and by 0 otherwise;
- the transfer instruction $T(m,n)$ replaces the contents of R_n by the natural number contained in R_m, leaving all other registers unaltered;
- the jump instruction $J(m,n,q)$ is carried out as follows: the contents r_m and r_n of the registers R_m and R_n are compared, all registers are left unaltered; then, if $r_m = r_n$, the URM proceeds to the qth instruction of P; if $r_m \neq r_n$,

URM-computability

How an URM works should now be clear: Simply run through the lines and act according to the commands.

A function $f : \omega \to \omega$ is called URM-computable iff there is a URM-program P that, starting with n in register R_1, stops after finitely many steps with $f(n)$ in register R_1.

A subset x of ω is computable if its characteristic function is. As usual, we identify $\mathbf{P}(\omega)$ with the real numbers.

Fact: Every URM-computable function is computable on a URM with 3 registers.

Register computations along an ordinal time axis

Keep the 'hardware': Finitely many registers, each can store an integer.

Also, keep the notion of a program and the way the computation works at successor steps.

At limit times, let $R_i(\lambda) = liminf_{\iota<\lambda} R_i(\iota)$ for each $i \in \omega$, if it exists, and $R_i(\lambda) = 0$, otherwise. Also, the index of the active program line $l(\lambda)$ is defined to be $liminf_{\iota<\lambda} l_\iota$.

These machines are called "Infinite Time Register Machines" (ITRMs), introduced by Koepke.

Register computations along an ordinal time axis

Keep the ’hardware’: Finitely many registers, each can store an integer.
Also, keep the notion of a program and the way the computation works at successor steps.

At limit times, let $R_i(\lambda) = \mathit{liminf}_{\iota<\lambda} R_i(\iota)$ for each $i \in \omega$, if it exists, and $R_i(\lambda) = 0$, otherwise. Also, the index of the active program line $l(\lambda)$ is defined to be $\mathit{liminf}_{\iota<\lambda} l_\iota$.

These machines are called “Infinite Time Register Machines” (ITRMs), introduced by Koepke.

Register computations along an ordinal time axis

Keep the 'hardware': Finitely many registers, each can store an integer.
Also, keep the notion of a program and the way the computation works at successor steps.

At limit times, let $R_i(\lambda) = \mathit{liminf}_{\iota<\lambda} R_i(\iota)$ for each $i \in \omega$, if it exists, and $R_i(\lambda) = 0$, otherwise. Also, the index of the active program line $l(\lambda)$ is defined to be $\mathit{liminf}_{\iota<\lambda} l_\iota$.

These machines are called "Infinite Time Register Machines" (ITRMs), introduced by Koepke.

Register computations along an ordinal time axis

Keep the 'hardware': Finitely many registers, each can store an integer.
Also, keep the notion of a program and the way the computation works at successor steps.

At limit times, let $R_i(\lambda) = \mathit{liminf}_{\iota<\lambda} R_i(\iota)$ for each $i \in \omega$, if it exists, and $R_i(\lambda) = 0$, otherwise. Also, the index of the active program line $l(\lambda)$ is defined to be $\mathit{liminf}_{\iota<\lambda} l_\iota$.

These machines are called "Infinite Time Register Machines" (ITRMs), introduced by Koepke.

Theorem (Koepke/Miller): There is an ITRM-program P that decides WO, the set of real numbers that code well-orderings.
As a consequence, every Π^1_1-set is ITRM-decidable.

Theorem (Koepke): A real number x is ITRM-computable if and only if $x \in L_{\omega_\omega^{\mathrm{CK}}}$.

Theorem (Koepke/Miller): There is an ITRM-program *P* that decides WO, the set of real numbers that code well-orderings. As a consequence, every Π^1_1-set is ITRM-decidable.

Theorem (Koepke): A real number x is ITRM-computable if and only if $x \in L_{\omega_\omega^{CK}}$.

Theorem (Koepke/Miller): There is an ITRM-program P that decides WO, the set of real numbers that code well-orderings. As a consequence, every Π^1_1-set is ITRM-decidable.

Theorem (Koepke): A real number x is ITRM-computable if and only if $x \in L_{\omega_\omega^{CK}}$.

Infinite Time Turing Machines

*ITTM*s have the same 'hardware' as Turing machines: They have a tape with cells indexed with natural numbers (each of which can contain a 0 or a 1), a read/write head, a finite set of internal states, represented by natural numbers and possibly an oracle.
They also have the same 'software': Commands that, depending on the current state and the symbol currently read, tell the machine what symbol to write, which new internal state to assume and where to move the read/write head.
However, the working time of an *ITTM* can be an arbitrary ordinal.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps.

But now, what should the state of the machine be at a limit time λ?

The internal state s_λ at time λ, we set $s_\lambda := \liminf\{s_\iota | \iota < \lambda\}$.

For *ITTMs*, the head position p_λ at time λ is $p_\lambda := \liminf\{p_\iota | \iota < \lambda\}$, if this limit is finite; otherwise, we set $p_\lambda = 0$.

Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \liminf\{t_{\iota\gamma} | \gamma < \lambda\}$.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps.
But now, what should the state of the machine be at a limit time λ?
The internal state s_λ at time λ, we set $s_\lambda := \liminf\{s_\iota | \iota < \lambda\}$.
For *ITTMs*, the head position p_λ at time λ is $p_\lambda := \liminf\{p_\iota | \iota < \lambda\}$, if this limit is finite; otherwise, we set $p_\lambda = 0$.
Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \liminf\{t_{\iota\gamma} | \gamma < \lambda\}$.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps.

But now, what should the state of the machine be at a limit time λ?

The internal state s_λ at time λ, we set $s_\lambda := \text{liminf}\{s_\iota | \iota < \lambda\}$.

For *ITTMs*, the head position p_λ at time λ is $p_\lambda := \text{liminf}\{p_\iota | \iota < \lambda\}$, if this limit is finite; otherwise, we set $p_\lambda = 0$.

Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \text{liminf}\{t_{\iota\gamma} | \gamma < \lambda\}$.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps.
But now, what should the state of the machine be at a limit time λ?
The internal state s_λ at time λ, we set $s_\lambda := \liminf\{s_\iota | \iota < \lambda\}$.
For *ITTMs*, the head position p_λ at time λ is $p_\lambda := \liminf\{p_\iota | \iota < \lambda\}$, if this limit is finite; otherwise, we set $p_\lambda = 0$.
Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \liminf\{t_{\iota\gamma} | \gamma < \lambda\}$.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps.
But now, what should the state of the machine be at a limit time λ?
The internal state s_λ at time λ, we set $s_\lambda := \liminf\{s_\iota | \iota < \lambda\}$.
For *ITTMs*, the head position p_λ at time λ is $p_\lambda := \liminf\{p_\iota | \iota < \lambda\}$, if this limit is finite; otherwise, we set $p_\lambda = 0$.
Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \liminf\{t_{\iota\gamma} | \gamma < \lambda\}$.

ITTM-computability

For *ITTMs*, we have three important notions of writability, due to Hamkins and Lewis:

$x \subseteq \omega$ is ITTM-writable iff there is an ITTM-program P that, when run on the empty tape, halts with x on the tape.

$x \subseteq \omega$ is eventually ITTM-writable iff there is an ITTM-program P such that, when P is run on the empty tape, there is for every $n \in \omega$ an ordinal α such that the nth cell of the tape contains 1 iff $n \in x$ from time α on.
(Think: Tape content 'converges to' x)

$x \subseteq \omega$ is accidentally ITTM-writable iff there are an ITTM-program P and an ordinal α such that the tape content at time α is x when P is run on the empty tape.

Fact: (Hamkins/Lewis) x writable $\rightarrow$ x eventually writable $\rightarrow$ x accidentally writable. None of these implications can be reversed.

All of these notions and the fact relativize to oracles.

ITTM-computability

For *ITTM*s, we have three important notions of writability, due to Hamkins and Lewis:

$x \subseteq \omega$ is ITTM-writable iff there is an ITTM-program P that, when run on the empty tape, halts with x on the tape.

$x \subseteq \omega$ is eventually ITTM-writable iff there is an ITTM-program P such that, when P is run on the empty tape, there is for every $n \in \omega$ an ordinal α such that the nth cell of the tape contains 1 iff $n \in x$ from time α on.

(Think: Tape content 'converges to' x)

$x \subseteq \omega$ is accidentally ITTM-writable iff there are an ITTM-program P and an ordinal α such that the tape content at time α is x when P is run on the empty tape.

Fact: (Hamkins/Lewis) x writable $\rightarrow$ x eventually writable $\rightarrow$ x accidentally writable. None of these implications can be reversed.

All of these notions and the fact relativize to oracles.

ITTM-computability

For *ITTM*s, we have three important notions of writability, due to Hamkins and Lewis:

$x \subseteq \omega$ is ITTM-writable iff there is an ITTM-program P that, when run on the empty tape, halts with x on the tape.

$x \subseteq \omega$ is eventually ITTM-writable iff there is an ITTM-program P such that, when P is run on the empty tape, there is for every $n \in \omega$ an ordinal α such that the nth cell of the tape contains 1 iff $n \in x$ from time α on.
(Think: Tape content 'converges to' x)

$x \subseteq \omega$ is accidentally ITTM-writable iff there are an ITTM-program P and an ordinal α such that the tape content at time α is x when P is run on the empty tape.

Fact: (Hamkins/Lewis) x writable $\rightarrow$ x eventually writable $\rightarrow$ x accidentally writable. None of these implications can be reversed.

All of these notions and the fact relativize to oracles.

ITTM-computability

For *ITTM*s, we have three important notions of writability, due to Hamkins and Lewis:

$x \subseteq \omega$ is ITTM-writable iff there is an ITTM-program P that, when run on the empty tape, halts with x on the tape.

$x \subseteq \omega$ is eventually ITTM-writable iff there is an ITTM-program P such that, when P is run on the empty tape, there is for every $n \in \omega$ an ordinal α such that the nth cell of the tape contains 1 iff $n \in x$ from time α on.

(Think: Tape content 'converges to' x)

$x \subseteq \omega$ is accidentally ITTM-writable iff there are an ITTM-program P and an ordinal α such that the tape content at time α is x when P is run on the empty tape.

Fact: (Hamkins/Lewis) x writable $\rightarrow$ x eventually writable $\rightarrow$ x accidentally writable. None of these implications can be reversed.

All of these notions and the fact relativize to oracles.

ITTM-computability

For *ITTM*s, we have three important notions of writability, due to Hamkins and Lewis:

$x \subseteq \omega$ is ITTM-writable iff there is an ITTM-program P that, when run on the empty tape, halts with x on the tape.

$x \subseteq \omega$ is eventually ITTM-writable iff there is an ITTM-program P such that, when P is run on the empty tape, there is for every $n \in \omega$ an ordinal α such that the nth cell of the tape contains 1 iff $n \in x$ from time α on.

(Think: Tape content 'converges to' x)

$x \subseteq \omega$ is accidentally ITTM-writable iff there are an ITTM-program P and an ordinal α such that the tape content at time α is x when P is run on the empty tape.

Fact: (Hamkins/Lewis) x writable $\rightarrow$ x eventually writable $\rightarrow$ x accidentally writable. None of these implications can be reversed.

All of these notions and the fact relativize to oracles.

ITTM-computability

For *ITTM*s, we have three important notions of writability, due to Hamkins and Lewis:

$x \subseteq \omega$ is ITTM-writable iff there is an ITTM-program P that, when run on the empty tape, halts with x on the tape.

$x \subseteq \omega$ is eventually ITTM-writable iff there is an ITTM-program P such that, when P is run on the empty tape, there is for every $n \in \omega$ an ordinal α such that the nth cell of the tape contains 1 iff $n \in x$ from time α on.

(Think: Tape content 'converges to' x)

$x \subseteq \omega$ is accidentally ITTM-writable iff there are an ITTM-program P and an ordinal α such that the tape content at time α is x when P is run on the empty tape.

Fact: (Hamkins/Lewis) x writable $\rightarrow$ x eventually writable $\rightarrow$ x accidentally writable. None of these implications can be reversed.

All of these notions and the fact relativize to oracles.

A simple ITTM-computation

$(q_0, 0) \rightarrow (q_0, 1, \text{right})$
$(q_0, 1) \rightarrow (q_0, 0, \text{right})$

Time	State	Cell 0	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	...
0	q_0	0	0	0	0	0	0	...
1	q_0	1	0	0	0	0	0	...
2	q_0	1	1	0	0	0	0	...
...	...	...	...	...	...	...	...	...
ω	q_0	1	1	1	1	1	1	...
$\omega+1$	q_0	0	1	1	1	1	1	...
$\omega+2$	q_0	0	0	1	1	1	1	...
...	...	...	...	...	...	...	...	...
$\omega \cdot 2$	q_0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
ω^2	q_0	0	0	0	0	0	0	...

A simple ITTM-computation

$(q_0, 0) \rightarrow (q_0, 1, \text{right})$
$(q_0, 1) \rightarrow (q_0, 0, \text{right})$

Time	State	Cell 0	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	...
0	q_0	0	0	0	0	0	0	...
1	q_0	1	0	0	0	0	0	...
2	q_0	1	1	0	0	0	0	...
...	...	...	...	...	...	...	...	...
ω	q_0	1	1	1	1	1	1	...
$\omega+1$	q_0	0	1	1	1	1	1	...
$\omega+2$	q_0	0	0	1	1	1	1	...
...	...	...	...	...	...	...	...	...
$\omega \cdot 2$	q_0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
ω^2	q_0	0	0	0	0	0	0	...

A simple ITTM-computation

$(q_0, 0) \rightarrow (q_0, 1, \text{right})$
$(q_0, 1) \rightarrow (q_0, 0, \text{right})$

Time	State	Cell 0	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	...
0	q_0	0	0	0	0	0	0	...
1	q_0	1	0	0	0	0	0	...
2	q_0	1	1	0	0	0	0	...
...	...	...	...	...	...	...	...	...
ω	q_0	1	1	1	1	1	1	...
$\omega+1$	q_0	0	1	1	1	1	1	...
$\omega+2$	q_0	0	0	1	1	1	1	...
...	...	...	...	...	...	...	...	...
$\omega \cdot 2$	q_0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
ω^2	q_0	0	0	0	0	0	0	...

A simple ITTM-computation

$(q_0, 0) \to (q_0, 1, \text{right})$
$(q_0, 1) \to (q_0, 0, \text{right})$

Time	State	Cell 0	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	...
0	q_0	0	0	0	0	0	0	...
1	q_0	1	0	0	0	0	0	...
2	q_0	1	1	0	0	0	0	...
...	...	...	...	...	...	...	...	...
ω	q_0	1	1	1	1	1	1	...
$\omega + 1$	q_0	0	1	1	1	1	1	...
$\omega + 2$	q_0	0	0	1	1	1	1	...
...	...	...	...	...	...	...	...	...
$\omega \cdot 2$	q_0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
ω^2	q_0	0	0	0	0	0	0	...

A simple ITTM-computation

$(q_0, 0) \to (q_0, 1, \text{right})$
$(q_0, 1) \to (q_0, 0, \text{right})$

Time	State	Cell 0	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	...
0	q_0	0	0	0	0	0	0	...
1	q_0	1	0	0	0	0	0	...
2	q_0	1	1	0	0	0	0	...
...	...	...	...	...	...	...	...	...
ω	q_0	1	1	1	1	1	1	...
$\omega+1$	q_0	0	1	1	1	1	1	...
$\omega+2$	q_0	0	0	1	1	1	1	...
...	...	...	...	...	...	...	...	...
$\omega \cdot 2$	q_0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
ω^2	q_0	0	0	0	0	0	0	...

A simple ITTM-computation

$(q_0, 0) \to (q_0, 1, \text{right})$
$(q_0, 1) \to (q_0, 0, \text{right})$

Time	State	Cell 0	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	...
0	q_0	0	0	0	0	0	0	...
1	q_0	1	0	0	0	0	0	...
2	q_0	1	1	0	0	0	0	...
...	...	...	...	...	...	...	...	...
ω	q_0	1	1	1	1	1	1	...
$\omega + 1$	q_0	0	1	1	1	1	1	...
$\omega + 2$	q_0	0	0	1	1	1	1	...
...	...	...	...	...	...	...	...	...
$\omega \cdot 2$	q_0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
ω^2	q_0	0	0	0	0	0	0	...

A simple ITTM-computation

$(q_0, 0) \to (q_0, 1, \text{right})$
$(q_0, 1) \to (q_0, 0, \text{right})$

Time	State	Cell 0	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	...
0	q_0	0	0	0	0	0	0	...
1	q_0	1	0	0	0	0	0	...
2	q_0	1	1	0	0	0	0	...
...	...	...	...	...	...	...	...	...
ω	q_0	1	1	1	1	1	1	...
$\omega+1$	q_0	0	1	1	1	1	1	...
$\omega+2$	q_0	0	0	1	1	1	1	...
...	...	...	...	...	...	...	...	...
$\omega \cdot 2$	q_0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
ω^2	q_0	0	0	0	0	0	0	...

A simple ITTM-computation

$(q_0, 0) \rightarrow (q_0, 1, \text{right})$
$(q_0, 1) \rightarrow (q_0, 0, \text{right})$

Time	State	Cell 0	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	...
0	q_0	0	0	0	0	0	0	...
1	q_0	1	0	0	0	0	0	...
2	q_0	1	1	0	0	0	0	...
...	...	...	...	...	...	...	...	...
ω	q_0	1	1	1	1	1	1	...
$\omega + 1$	q_0	0	1	1	1	1	1	...
$\omega + 2$	q_0	0	0	1	1	1	1	...
...	...	...	...	...	...	...	...	...
$\omega \cdot 2$	q_0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
ω^2	q_0	0	0	0	0	0	0	...

Characterizing ITTM-computability

Theorem: (Welch) There are ordinals $\lambda < \zeta < \Sigma$ such that $x \subseteq \omega$ is writable/eventually writable/accidentally writable by an *ITTM* iff x is an element of L_λ, L_ζ, L_Σ, respectively. Moreover, (λ, ζ, Σ) can be characterized as the lexically minimal tuple (α, β, γ) of ordinals such that $L_\alpha \prec_{\Sigma_1} L_\beta \prec_{\Sigma_2} L_\gamma$.

The relativization to oracles holds as well.

Characterizing ITTM-computability

Theorem: (Welch) There are ordinals $\lambda < \zeta < \Sigma$ such that $x \subseteq \omega$ is writable/eventually writable/accidentally writable by an *ITTM* iff x is an element of L_λ, L_ζ, L_Σ, respectively. Moreover, (λ, ζ, Σ) can be characterized as the lexically minimal tuple (α, β, γ) of ordinals such that $L_\alpha \prec_{\Sigma_1} L_\beta \prec_{\Sigma_2} L_\gamma$.

The relativization to oracles holds as well.

Characterizing ITTM-computability

Theorem: (Welch) There are ordinals $\lambda < \zeta < \Sigma$ such that $x \subseteq \omega$ is writable/eventually writable/accidentally writable by an *ITTM* iff x is an element of L_λ, L_ζ, L_Σ, respectively. Moreover, (λ, ζ, Σ) can be characterized as the lexically minimal tuple (α, β, γ) of ordinals such that $L_\alpha \prec_{\Sigma_1} L_\beta \prec_{\Sigma_2} L_\gamma$.

The relativization to oracles holds as well.

Decidability

Let X be a set of real numbers.

X is called ITTM/ITRM-decidable if and only if there is an ITTM/ITRM-program P such that, for all $x \subseteq \omega$, $P^x \downarrow = 1$ if and only if $x \in X$ and otherwise $P^x \downarrow = 0$.

X is called ITTM-semidecidable if and only if there is an ITTM-program P such that, for all $x \subseteq \omega$, P^x halts if and only if $x \in X$.

X is called ITTM-cosemidecidable if and only if there is an ITTM-program P such that, for all $x \subseteq \omega$, P^x halts if and only if $x \notin X$.

Decidability

Let X be a set of real numbers.
X is called ITTM/ITRM-decidable if and only if there is an ITTM/ITRM-program P such that, for all $x \subseteq \omega$, $P^x \downarrow = 1$ if and only if $x \in X$ and otherwise $P^x \downarrow = 0$.

X is called ITTM-semidecidable if and only if there is an ITTM-program P such that, for all $x \subseteq \omega$, P^x halts if and only if $x \in X$.

X is called ITTM-cosemidecidable if and only if there is an ITTM-program P such that, for all $x \subseteq \omega$, P^x halts if and only if $x \notin X$.

Decidability

Let X be a set of real numbers.
X is called ITTM/ITRM-decidable if and only if there is an ITTM/ITRM-program P such that, for all $x \subseteq \omega$, $P^x \downarrow = 1$ if and only if $x \in X$ and otherwise $P^x \downarrow = 0$.

X is called ITTM-semidecidable if and only if there is an ITTM-program P such that, for all $x \subseteq \omega$, P^x halts if and only if $x \in X$.

X is called ITTM-cosemidecidable if and only if there is an ITTM-program P such that, for all $x \subseteq \omega$, P^x halts if and only if $x \notin X$.

Decidability

Let X be a set of real numbers.
X is called ITTM/ITRM-decidable if and only if there is an ITTM/ITRM-program P such that, for all $x \subseteq \omega$, $P^x \downarrow = 1$ if and only if $x \in X$ and otherwise $P^x \downarrow = 0$.

X is called ITTM-semidecidable if and only if there is an ITTM-program P such that, for all $x \subseteq \omega$, P^x halts if and only if $x \in X$.

X is called ITTM-cosemidecidable if and only if there is an ITTM-program P such that, for all $x \subseteq \omega$, P^x halts if and only if $x \notin X$.

Time Complexity for ITTMs

(Schindler)
A set X of real numbers (=0-1-strings of length ω, subsets of ω) is ITTM-decidable with time bound $f : \mathbb{R} \to \mathrm{On}$ if and only if there is an ITTM-program P that decides X and runs for $\leq f(x)$ many steps on input x.

Space Complexity for ITTMs

This is not quite as easy. An ITTM has a tape of length ω, and in all but the most trivial cases uses all of it.

Idea (Löwe): Rather than the total amount of cells used, measure the "complexity" of the occuring snapshots.

A set X of real number is of space complexity $f : \mathbb{R} \rightarrow \mathrm{On}$ if and only if there is an ITTM-program P that decides X and, on input x, only produces elements of $L_{\alpha}[x]$ on its scratch tape.
The class of these sets is denoted as $\mathrm{SPACE}^{\mathrm{ITTM}}_{\alpha}$.

Space Complexity for ITTMs

This is not quite as easy. An ITTM has a tape of length ω, and in all but the most trivial cases uses all of it.

Idea (Löwe): Rather than the total amount of cells used, measure the “complexity” of the occuring snapshots.

A set X of real number is of space complexity $f : \mathbb{R} \to \mathrm{On}$ if and only if there is an ITTM-program P that decides X and, on input x, only produces elements of $L_\alpha[x]$ on its scratch tape.

The class of these sets is denoted as $\mathrm{SPACE}_\alpha^{\mathrm{ITTM}}$.

Space Complexity for ITTMs

This is not quite as easy. An ITTM has a tape of length ω, and in all but the most trivial cases uses all of it.

Idea (Löwe): Rather than the total amount of cells used, measure the “complexity” of the occuring snapshots.

A set X of real number is of space complexity $f : \mathbb{R} \rightarrow \mathrm{On}$ if and only if there is an ITTM-program P that decides X and, on input x, only produces elements of $L_{\alpha}[x]$ on its scratch tape.

The class of these sets is denoted as $\mathrm{SPACE}_{\alpha}^{\mathrm{ITTM}}$.

Space Complexity for ITTMs

This is not quite as easy. An ITTM has a tape of length ω, and in all but the most trivial cases uses all of it.

Idea (Löwe): Rather than the total amount of cells used, measure the “complexity” of the occuring snapshots.

A set X of real number is of space complexity $f : \mathbb{R} \to$ On if and only if there is an ITTM-program P that decides X and, on input x, only produces elements of $L_\alpha[x]$ on its scratch tape.
The class of these sets is denoted as $\mathrm{SPACE}_\alpha^{\mathrm{ITTM}}$.

The Bold Conjecture

It is trivial that low time complexity implies low space complexity: In α many steps, you cannot step outside of $L_\alpha[x]$ on input x.

Löwe[1] asked whether there is a converse: Does low space complexity imply low time complexity? Are sets that are decidable with "simple" snapshots also "quickly" decidable?

In particular: If $X \subseteq \mathfrak{P}(\omega)$ is ITTM-decidable with recursive snapshots, does that imply that X is decidable with some uniform time bound $\gamma < \omega^\omega$ (which was Schindler's definition of P for ITTMs)?

[1] Space bounds for infinitary computation. (CiE 2006 Proceedings)

The Bold Conjecture

It is trivial that low time complexity implies low space complexity: In α many steps, you cannot step outside of $L_\alpha[x]$ on input x.

Löwe[1] asked whether there is a converse: Does low space complexity imply low time complexity? Are sets that are decidable with "simple" snapshots also "quickly" decidable?

In particular: If $X \subseteq \mathfrak{P}(\omega)$ is ITTM-decidable with recursive snapshots, does that imply that X is decidable with some uniform time bound $\gamma < \omega^\omega$ (which was Schindler's definition of P for ITTMs)?

[1] Space bounds for infinitary computation. (CiE 2006 Proceedings)

The Bold Conjecture

It is trivial that low time complexity implies low space complexity: In α many steps, you cannot step outside of $L_\alpha[x]$ on input x.

Löwe[1] asked whether there is a converse: Does low space complexity imply low time complexity? Are sets that are decidable with "simple" snapshots also "quickly" decidable?

In particular: If $X \subseteq \mathfrak{P}(\omega)$ is ITTM-decidable with recursive snapshots, does that imply that X is decidable with some uniform time bound $\gamma < \omega^\omega$ (which was Schindler's definition of P for ITTMs)?

[1] Space bounds for infinitary computation. (CiE 2006 Proceedings)

The Bold Conjecture

It is trivial that low time complexity implies low space complexity: In α many steps, you cannot step outside of $L_\alpha[x]$ on input x.

Löwe[1] asked whether there is a converse: Does low space complexity imply low time complexity? Are sets that are decidable with “simple” snapshots also “quickly” decidable?

In particular: If $X \subseteq \mathfrak{P}(\omega)$ is ITTM-decidable with recursive snapshots, does that imply that X is decidable with some uniform time bound $\gamma < \omega^\omega$ (which was Schindler's definition of P for ITTMs)?

[1] Space bounds for infinitary computation. (CiE 2006 Proceedings)

Nice Question. And now for something completely different

The concept of recognizability was defined by Hamkins and Lewis for ITTMs; it has no straightforward analogue in finite computability.

A real number x such that $\{x\}$ is ITTM-decidable is called ITTM-recognizable. (And similarly for ITRMs.)

Theorem: (Hamkins/Lewis, the "lost melody theorem" for ITTMs) There are real numbers x that are ITTM-recognizable, but not ITTM-writable.
The same holds for ITRMs. (C.)

Fact (Folklore): If x is ITTM-recognizable, then $x \in L_{\lambda^x}$. If x is ITRM-recognizable, then $x \in L_{\omega_\omega^{CK,x}}$.

Nice Question. And now for something completely different

The concept of recognizability was defined by Hamkins and Lewis for ITTMs; it has no straightforward analogue in finite computability.

A real number x such that $\{x\}$ is ITTM-decidable is called ITTM-recognizable. (And similarly for ITRMs.)

Theorem: (Hamkins/Lewis, the “lost melody theorem” for ITTMs) There are real numbers x that are ITTM-recognizable, but not ITTM-writable.
The same holds for ITRMs. (C.)

Fact (Folklore): If x is ITTM-recognizable, then $x \in L_{\lambda^x}$. If x is ITRM-recognizable, then $x \in L_{\omega_\omega^{CK,x}}$.

Nice Question. And now for something completely different

The concept of recognizability was defined by Hamkins and Lewis for ITTMs; it has no straightforward analogue in finite computability.

A real number x such that $\{x\}$ is ITTM-decidable is called ITTM-recognizable. (And similarly for ITRMs.)

Theorem: (Hamkins/Lewis, the "lost melody theorem" for ITTMs) There are real numbers x that are ITTM-recognizable, but not ITTM-writable.
The same holds for ITRMs. (C.)

Fact (Folklore): If x is ITTM-recognizable, then $x \in L_{\lambda^x}$. If x is ITRM-recognizable, then $x \in L_{\omega_\omega^{CK,x}}$.

Nice Question. And now for something completely different

The concept of recognizability was defined by Hamkins and Lewis for ITTMs; it has no straightforward analogue in finite computability.

A real number x such that $\{x\}$ is ITTM-decidable is called ITTM-recognizable. (And similarly for ITRMs.)

Theorem: (Hamkins/Lewis, the "lost melody theorem" for ITTMs) There are real numbers x that are ITTM-recognizable, but not ITTM-writable.
The same holds for ITRMs. (C.)

Fact (Folklore): If x is ITTM-recognizable, then $x \in L_{\lambda^x}$. If x is ITRM-recognizable, then $x \in L_{\omega_\omega^{CK,x}}$.

Nice Question. And now for something completely different

The concept of recognizability was defined by Hamkins and Lewis for ITTMs; it has no straightforward analogue in finite computability.

A real number x such that $\{x\}$ is ITTM-decidable is called ITTM-recognizable. (And similarly for ITRMs.)

Theorem: (Hamkins/Lewis, the "lost melody theorem" for ITTMs) There are real numbers x that are ITTM-recognizable, but not ITTM-writable.
The same holds for ITRMs. (C.)

Fact (Folklore): If x is ITTM-recognizable, then $x \in L_{\lambda^x}$. If x is ITRM-recognizable, then $x \in L_{\omega_\omega^{CK,x}}$.

Nice Question. And now for something completely different

The concept of recognizability was defined by Hamkins and Lewis for ITTMs; it has no straightforward analogue in finite computability.

A real number x such that $\{x\}$ is ITTM-decidable is called ITTM-recognizable. (And similarly for ITRMs.)

Theorem: (Hamkins/Lewis, the “lost melody theorem” for ITTMs) There are real numbers x that are ITTM-recognizable, but not ITTM-writable.
The same holds for ITRMs. (C.)

Fact (Folklore): If x is ITTM-recognizable, then $x \in L_{\lambda^x}$. If x is ITRM-recognizable, then $x \in L_{\omega_\omega^{CK,x}}$.

Nice Question. And now for something completely different

The concept of recognizability was defined by Hamkins and Lewis for ITTMs; it has no straightforward analogue in finite computability.

A real number x such that $\{x\}$ is ITTM-decidable is called ITTM-recognizable. (And similarly for ITRMs.)

Theorem: (Hamkins/Lewis, the "lost melody theorem" for ITTMs) There are real numbers x that are ITTM-recognizable, but not ITTM-writable.
The same holds for ITRMs. (C.)

Fact (Folklore): If x is ITTM-recognizable, then $x \in L_{\lambda^x}$. If x is ITRM-recognizable, then $x \in L_{\omega_\omega^{\mathrm{CK},x}}$.

Some basic facts about ITRM-recognizability

Let us denote by σ the minimal ordinal satisfying $L_\sigma \prec_{\Sigma_1} L$.

Equivalently, σ is the supremum of the $\Sigma_1^{L_{\omega_1}}$-definable ordinals.

Then σ is also minimal with the property that L_σ contains all ITRM-recognizable real numbers (the same is true for ITTMs).

In fact, if $\alpha < \sigma$ is minimal with the property that $L_\alpha \models \phi$ for some $\in$-sentence ϕ, then the $<_L$-minimal real number c that codes L_α is ITRM-recognizable.

Some basic facts about ITRM-recognizability

Let us denote by σ the minimal ordinal satisfying $L_\sigma \prec_{\Sigma_1} L$. Equivalently, σ is the supremum of the $\Sigma_1^{L_{\omega_1}}$-definable ordinals.

Then σ is also minimal with the property that L_σ contains all ITRM-recognizable real numbers (the same is true for ITTMs).

In fact, if $\alpha < \sigma$ is minimal with the property that $L_\alpha \models \phi$ for some $\in$-sentence ϕ, then the $<_L$-minimal real number c that codes L_α is ITRM-recognizable.

Some basic facts about ITRM-recognizability

Let us denote by σ the minimal ordinal satisfying $L_\sigma \prec_{\Sigma_1} L$. Equivalently, σ is the supremum of the $\Sigma_1^{L_{\omega_1}}$-definable ordinals.

Then σ is also minimal with the property that L_σ contains all ITRM-recognizable real numbers (the same is true for ITTMs).

In fact, if $\alpha < \sigma$ is minimal with the property that $L_\alpha \models \phi$ for some $\in$-sentence ϕ, then the $<_L$-minimal real number c that codes L_α is ITRM-recognizable.

Some basic facts about ITRM-recognizability

Let us denote by σ the minimal ordinal satisfying $L_\sigma \prec_{\Sigma_1} L$. Equivalently, σ is the supremum of the $\Sigma_1^{L_{\omega_1}}$-definable ordinals.

Then σ is also minimal with the property that L_σ contains all ITRM-recognizable real numbers (the same is true for ITTMs).

In fact, if $\alpha < \sigma$ is minimal with the property that $L_\alpha \models \phi$ for some $\in$-sentence ϕ, then the $<_L$-minimal real number c that codes L_α is ITRM-recognizable.

Some basic facts about ITRM-recognizability

Let us denote by σ the minimal ordinal satisfying $L_\sigma \prec_{\Sigma_1} L$. Equivalently, σ is the supremum of the $\Sigma_1^{L_{\omega_1}}$-definable ordinals.

Then σ is also minimal with the property that L_σ contains all ITRM-recognizable real numbers (the same is true for ITTMs).

In fact, if $\alpha < \sigma$ is minimal with the property that $L_\alpha \models \phi$ for some $\in$-sentence ϕ, then the $<_L$-minimal real number c that codes L_α is ITRM-recognizable.

ITRMs refute the bold conjecture

These results about ITRM-recognizability seem unrelated to the bold conjecture. However, they can easily be used to show that it is false:

- An ITRM with n registers can be simulated by an ITTM with n scratch tapes by representing the content $n \in \omega$ of register R_i on the i-th scratch tape by n many 1s followed by 0s.
- The snapshots occuring thereby are clearly recursive, and in fact as simple as one could possibly ask for.
- But this means that ITTMs with extremely simple snapshots can decide every ITRM-decidable set, such as WO.
- However, WO is Π^1_1-universal, and every set that is ITTM-decidable with a constant countable space-bound γ is Σ^1_1 in any code for γ.
- Thus, we have a set that is not ITTM-decidable with any countable time bound (the minimal time bound for deciding WO is ω_1), but ITTM-decidable with extremely simple snapshots.

ITRMs refute the bold conjecture

These results about ITRM-recognizability seem unrelated to the bold conjecture. However, they can easily be used to show that it is false:

- An ITRM with n registers can be simulated by an ITTM with n scratch tapes by representing the content $n \in \omega$ of register R_i on the i-th scratch tape by n many 1s followed by 0s.
- The snapshots occuring thereby are clearly recursive, and in fact as simple as one could possibly ask for.
- But this means that ITTMs with extremely simple snapshots can decide every ITRM-decidable set, such as WO.
- However, WO is Π^1_1-universal, and every set that is ITTM-decidable with a constant countable space-bound γ is Σ^1_1 in any code for γ.
- Thus, we have a set that is not ITTM-decidable with any countable time bound (the minimal time bound for deciding WO is ω_1), but ITTM-decidable with extremely simple snapshots.

ITRMs refute the bold conjecture

These results about ITRM-recognizability seem unrelated to the bold conjecture. However, they can easily be used to show that it is false:

- An ITRM with n registers can be simulated by an ITTM with n scratch tapes by representing the content $n \in \omega$ of register R_i on the i-th scratch tape by n many 1s followed by 0s.
- The snapshots occuring thereby are clearly recursive, and in fact as simple as one could possibly ask for.
- But this means that ITTMs with extremely simple snapshots can decide every ITRM-decidable set, such as WO.
- However, WO is Π^1_1-universal, and every set that is ITTM-decidable with a constant countable space-bound γ is Σ^1_1 in any code for γ.
- Thus, we have a set that is not ITTM-decidable with any countable time bound (the minimal time bound for deciding WO is ω_1), but ITTM-decidable with extremely simple snapshots.

ITRMs refute the bold conjecture

These results about ITRM-recognizability seem unrelated to the bold conjecture. However, they can easily be used to show that it is false:

- An ITRM with n registers can be simulated by an ITTM with n scratch tapes by representing the content $n \in \omega$ of register R_i on the i-th scratch tape by n many 1s followed by 0s.
- The snapshots occuring thereby are clearly recursive, and in fact as simple as one could possibly ask for.
- But this means that ITTMs with extremely simple snapshots can decide every ITRM-decidable set, such as WO.
- However, WO is Π^1_1-universal, and every set that is ITTM-decidable with a constant countable space-bound γ is Σ^1_1 in any code for γ.
- Thus, we have a set that is not ITTM-decidable with any countable time bound (the minimal time bound for deciding WO is ω_1), but ITTM-decidable with extremely simple snapshots.

ITRMs refute the bold conjecture

These results about ITRM-recognizability seem unrelated to the bold conjecture. However, they can easily be used to show that it is false:

- An ITRM with n registers can be simulated by an ITTM with n scratch tapes by representing the content $n \in \omega$ of register R_i on the i-th scratch tape by n many 1s followed by 0s.
- The snapshots occuring thereby are clearly recursive, and in fact as simple as one could possibly ask for.
- But this means that ITTMs with extremely simple snapshots can decide every ITRM-decidable set, such as WO.
- However, WO is Π^1_1-universal, and every set that is ITTM-decidable with a constant countable space-bound γ is Σ^1_1 in any code for γ.
- Thus, we have a set that is not ITTM-decidable with any countable time bound (the minimal time bound for deciding WO is ω_1), but ITTM-decidable with extremely simple snapshots.

ITRMs refute the bold conjecture

These results about ITRM-recognizability seem unrelated to the bold conjecture. However, they can easily be used to show that it is false:

- An ITRM with n registers can be simulated by an ITTM with n scratch tapes by representing the content $n \in \omega$ of register R_i on the i-th scratch tape by n many 1s followed by 0s.
- The snapshots occuring thereby are clearly recursive, and in fact as simple as one could possibly ask for.
- But this means that ITTMs with extremely simple snapshots can decide every ITRM-decidable set, such as WO.
- However, WO is Π^1_1-universal, and every set that is ITTM-decidable with a constant countable space-bound γ is Σ^1_1 in any code for γ.
- Thus, we have a set that is not ITTM-decidable with any countable time bound (the minimal time bound for deciding WO is ω_1), but ITTM-decidable with extremely simple snapshots.

ITRMs refute the bold conjecture

These results about ITRM-recognizability seem unrelated to the bold conjecture. However, they can easily be used to show that it is false:

- An ITRM with n registers can be simulated by an ITTM with n scratch tapes by representing the content $n \in \omega$ of register R_i on the i-th scratch tape by n many 1s followed by 0s.
- The snapshots occuring thereby are clearly recursive, and in fact as simple as one could possibly ask for.
- But this means that ITTMs with extremely simple snapshots can decide every ITRM-decidable set, such as WO.
- However, WO is Π^1_1-universal, and every set that is ITTM-decidable with a constant countable space-bound γ is Σ^1_1 in any code for γ.
- Thus, we have a set that is not ITTM-decidable with any countable time bound (the minimal time bound for deciding WO is ω_1), but ITTM-decidable with extremely simple snapshots.

ITRMs refute the bold conjecture

These results about ITRM-recognizability seem unrelated to the bold conjecture. However, they can easily be used to show that it is false:

- An ITRM with n registers can be simulated by an ITTM with n scratch tapes by representing the content $n \in \omega$ of register R_i on the i-th scratch tape by n many 1s followed by 0s.
- The snapshots occuring thereby are clearly recursive, and in fact as simple as one could possibly ask for.
- But this means that ITTMs with extremely simple snapshots can decide every ITRM-decidable set, such as WO.
- However, WO is Π^1_1-universal, and every set that is ITTM-decidable with a constant countable space-bound γ is Σ^1_1 in any code for γ.
- Thus, we have a set that is not ITTM-decidable with any countable time bound (the minimal time bound for deciding WO is ω_1), but ITTM-decidable with extremely simple snapshots.

ITRMs refute the bold conjecture

These results about ITRM-recognizability seem unrelated to the bold conjecture. However, they can easily be used to show that it is false:

- An ITRM with n registers can be simulated by an ITTM with n scratch tapes by representing the content $n \in \omega$ of register R_i on the i-th scratch tape by n many 1s followed by 0s.
- The snapshots occuring thereby are clearly recursive, and in fact as simple as one could possibly ask for.
- But this means that ITTMs with extremely simple snapshots can decide every ITRM-decidable set, such as WO.
- However, WO is Π^1_1-universal, and every set that is ITTM-decidable with a constant countable space-bound γ is Σ^1_1 in any code for γ.
- Thus, we have a set that is not ITTM-decidable with any countable time bound (the minimal time bound for deciding WO is ω_1), but ITTM-decidable with extremely simple snapshots.

Is there anything you can't do with recursive snapshots?

We consider a "proof" that ITTMs can solve their own halting problem. It will, of course, be false, but lead us into the right direction. Here it goes:

Fact (Hamkins/Lewis): An ITTM-program either halts or runs into a "strong loop"; that is a configuration c reappears, and in between the two occurences, all configurations majorized c component-wise.

So, in order to determine whether an ITTM-program P halts, we simply run P and keep track of the occuring snapshots. Whenever a snapshot occurs that is below some of the stored snapshots in at least one component, these snapshots are deleted and the new one is added. When a snapshot occurs that is already stored, we know that P never halts.

Is there anything you can't do with recursive snapshots?

We consider a "proof" that ITTMs can solve their own halting problem. It will, of course, be false, but lead us into the right direction. Here it goes:

Fact (Hamkins/Lewis): An ITTM-program either halts or runs into a "strong loop"; that is a configuration c reappears, and in between the two occurences, all configurations majorized c component-wise.

So, in order to determine whether an ITTM-program P halts, we simply run P and keep track of the occuring snapshots. Whenever a snapshot occurs that is below some of the stored snapshots in at least one component, these snapshots are deleted and the new one is added. When a snapshot occurs that is already stored, we know that P never halts.

Is there anything you can't do with recursive snapshots?

We consider a "proof" that ITTMs can solve their own halting problem. It will, of course, be false, but lead us into the right direction. Here it goes:

Fact (Hamkins/Lewis): An ITTM-program either halts or runs into a "strong loop"; that is a configuration c reappears, and in between the two occurences, all configurations majorized c component-wise.

So, in order to determine whether an ITTM-program P halts, we simply run P and keep track of the occuring snapshots. Whenever a snapshot occurs that is below some of the stored snapshots in at least one component, these snapshots are deleted and the new one is added. When a snapshot occurs that is already stored, we know that P never halts.

Is there anything you can't do with recursive snapshots?

We consider a "proof" that ITTMs can solve their own halting problem. It will, of course, be false, but lead us into the right direction. Here it goes:

Fact (Hamkins/Lewis): An ITTM-program either halts or runs into a "strong loop"; that is a configuration c reappears, and in between the two occurences, all configurations majorized c component-wise.

So, in order to determine whether an ITTM-program P halts, we simply run P and keep track of the occuring snapshots. Whenever a snapshot occurs that is below some of the stored snapshots in at least one component, these snapshots are deleted and the new one is added. When a snapshot occurs that is already stored, we know that P never halts.

Is there anything you can't do with recursive snapshots?

We consider a "proof" that ITTMs can solve their own halting problem. It will, of course, be false, but lead us into the right direction. Here it goes:

Fact (Hamkins/Lewis): An ITTM-program either halts or runs into a "strong loop"; that is a configuration c reappears, and in between the two occurences, all configurations majorized c component-wise.

So, in order to determine whether an ITTM-program P halts, we simply run P and keep track of the occuring snapshots. Whenever a snapshot occurs that is below some of the stored snapshots in at least one component, these snapshots are deleted and the new one is added. When a snapshot occurs that is already stored, we know that P never halts.

Is there anything you can't do with recursive snapshots?

We consider a "proof" that ITTMs can solve their own halting problem. It will, of course, be false, but lead us into the right direction. Here it goes:

Fact (Hamkins/Lewis): An ITTM-program either halts or runs into a "strong loop"; that is a configuration c reappears, and in between the two occurences, all configurations majorized c component-wise.

So, in order to determine whether an ITTM-program P halts, we simply run P and keep track of the occuring snapshots. Whenever a snapshot occurs that is below some of the stored snapshots in at least one component, these snapshots are deleted and the new one is added. When a snapshot occurs that is already stored, we know that P never halts.

Is there anything you can't do with recursive snapshots?

We consider a "proof" that ITTMs can solve their own halting problem. It will, of course, be false, but lead us into the right direction. Here it goes:

Fact (Hamkins/Lewis): An ITTM-program either halts or runs into a "strong loop"; that is a configuration c reappears, and in between the two occurences, all configurations majorized c component-wise.

So, in order to determine whether an ITTM-program P halts, we simply run P and keep track of the occuring snapshots. Whenever a snapshot occurs that is below some of the stored snapshots in at least one component, these snapshots are deleted and the new one is added. When a snapshot occurs that is already stored, we know that P never halts.

The reason why this does not work is that we cannot know beforehand how many snapshots will occur, so that we cannot organize the tape in order to "store" them in a way compatible with limits.

However, when all snapshots are recursive, we can simply store a snapshot s by finding a program P_i that (classically) computes s and marking i on the tape.

Thus, we can solve the halting problem for ITTMs with recursive snapshots on an ITTM.

A very similar argument shows that, for $\alpha < \lambda$, $\text{SPACE}_{\alpha}^{\text{ITTM}}$ is a proper subset of the set of ITTM-decidable sets.

The reason why this does not work is that we cannot know beforehand how many snapshots will occur, so that we cannot organize the tape in order to “store” them in a way compatible with limits.

However, when all snapshots are recursive, we can simply store a snapshot s by finding a program P_i that (classically) computes s and marking i on the tape.

Thus, we can solve the halting problem for ITTMs with recursive snapshots on an ITTM.

A very similar argument shows that, for $\alpha < \lambda$, $\text{SPACE}_{\alpha}^{\text{ITTM}}$ is a proper subset of the set of ITTM-decidable sets.

The reason why this does not work is that we cannot know beforehand how many snapshots will occur, so that we cannot organize the tape in order to "store" them in a way compatible with limits.

However, when all snapshots are recursive, we can simply store a snapshot s by finding a program P_i that (classically) computes s and marking i on the tape.

Thus, we can solve the halting problem for ITTMs with recursive snapshots on an ITTM.

A very similar argument shows that, for $\alpha < \lambda$, $\text{SPACE}_{\alpha}^{\text{ITTM}}$ is a proper subset of the set of ITTM-decidable sets.

The reason why this does not work is that we cannot know beforehand how many snapshots will occur, so that we cannot organize the tape in order to “store” them in a way compatible with limits.

However, when all snapshots are recursive, we can simply store a snapshot *s* by finding a program P_i that (classically) computes *s* and marking *i* on the tape.

Thus, we can solve the halting problem for ITTMs with recursive snapshots on an ITTM.

A very similar argument shows that, for $\alpha < \lambda$, $\mathrm{SPACE}_{\alpha}^{\mathrm{ITTM}}$ is a proper subset of the set of ITTM-decidable sets.

With some further analysis, we obtain:

- If $\alpha < \lambda$, then $\text{SPACE}_{\alpha}^{\text{ITTM}} \subsetneq \text{SPACE}_{\lambda}^{\text{ITTM}}$.
- If $\alpha < \lambda$, there is $\beta \in (\alpha, \lambda)$ such that $\text{SPACE}_{\alpha}^{\text{ITTM}} \subsetneq \text{SPACE}_{\beta}^{\text{ITTM}}$.
- There are cofinally many $\alpha < \sigma$ such that $\text{SPACE}_{\beta}^{\text{ITTM}} \subsetneq \text{SPACE}_{\alpha}^{\text{ITTM}}$ for all $\beta < \alpha$.

With some further analysis, we obtain:

- If $\alpha < \lambda$, then $\mathrm{SPACE}_{\alpha}^{\mathrm{ITTM}} \subsetneq \mathrm{SPACE}_{\lambda}^{\mathrm{ITTM}}$.
- If $\alpha < \lambda$, there is $\beta \in (\alpha, \lambda)$ such that $\mathrm{SPACE}_{\alpha}^{\mathrm{ITTM}} \subsetneq \mathrm{SPACE}_{\beta}^{\mathrm{ITTM}}$.
- There are cofinally many $\alpha < \sigma$ such that $\mathrm{SPACE}_{\beta}^{\mathrm{ITTM}} \subsetneq \mathrm{SPACE}_{\alpha}^{\mathrm{ITTM}}$ for all $\beta < \alpha$.

With some further analysis, we obtain:

- If $\alpha < \lambda$, then $\mathrm{SPACE}^{\mathrm{ITTM}}_{\alpha} \subsetneq \mathrm{SPACE}^{\mathrm{ITTM}}_{\lambda}$.
- If $\alpha < \lambda$, there is $\beta \in (\alpha, \lambda)$ such that $\mathrm{SPACE}^{\mathrm{ITTM}}_{\alpha} \subsetneq \mathrm{SPACE}^{\mathrm{ITTM}}_{\beta}$.
- There are cofinally many $\alpha < \sigma$ such that $\mathrm{SPACE}^{\mathrm{ITTM}}_{\beta} \subsetneq \mathrm{SPACE}^{\mathrm{ITTM}}_{\alpha}$ for all $\beta < \alpha$.

With some further analysis, we obtain:

- If $\alpha < \lambda$, then $\text{SPACE}^{\text{ITTM}}_{\alpha} \subsetneq \text{SPACE}^{\text{ITTM}}_{\lambda}$.
- If $\alpha < \lambda$, there is $\beta \in (\alpha, \lambda)$ such that $\text{SPACE}^{\text{ITTM}}_{\alpha} \subsetneq \text{SPACE}^{\text{ITTM}}_{\beta}$.
- There are cofinally many $\alpha < \sigma$ such that $\text{SPACE}^{\text{ITTM}}_{\beta} \subsetneq \text{SPACE}^{\text{ITTM}}_{\alpha}$ for all $\beta < \alpha$.

Aren't you playing unfair?

The WO-example suggests the following modification of the bold conjecture:

If X is ITTM-decidable with some fixed countable time bound α and also ITTM-decidable with recursive snapshots, then $\alpha = \omega^{\omega}$ (or at least, α should be "small" in some sense).

Aren't you playing unfair?

The WO-example suggests the following modification of the bold conjecture:

If X is ITTM-decidable with some fixed countable time bound α and also ITTM-decidable with recursive snapshots, then $\alpha = \omega^{\omega}$ (or at least, α should be "small" in some sense).

Aren't you playing unfair?

The WO-example suggests the following modification of the bold conjecture:

If X is ITTM-decidable with some fixed countable time bound α and also ITTM-decidable with recursive snapshots, then $\alpha = \omega^{\omega}$ (or at least, α should be “small” in some sense).

ITRMs refute the modified bold conjecture

Let $\alpha < \sigma$ be given. Pick $\beta < \sigma$ such that $\beta > \alpha^{+\omega}$ (the next limit of admissible ordinals after α) and β is minimal with the property that $L_\beta \models \phi$ for some $\in$-sentence ϕ.

Let c be the $<_L$-minimal real code for L_β.

By the results on ITRM-recognizability, c is ITRM-recognizable.

Thus, $\{c\}$ is ITTM-decidable with recursive (in fact, extremely simple) snapshots.

ITRMs refute the modified bold conjecture

Let $\alpha < \sigma$ be given. Pick $\beta < \sigma$ such that $\beta > \alpha^{+\omega}$ (the next limit of admissible ordinals after α) and β is minimal with the property that $L_\beta \models \phi$ for some $\in$-sentence ϕ.
Let c be the $<_L$-minimal real code for L_β.
By the results on ITRM-recognizability, c is ITRM-recognizable.
Thus, $\{c\}$ is ITTM-decidable with recursive (in fact, extremely simple) snapshots.

ITRMs refute the modified bold conjecture

Let $\alpha < \sigma$ be given. Pick $\beta < \sigma$ such that $\beta > \alpha^{+\omega}$ (the next limit of admissible ordinals after α) and β is minimal with the property that $L_\beta \models \phi$ for some $\in$-sentence ϕ.
Let c be the $<_L$-minimal real code for L_β.
By the results on ITRM-recognizability, c is ITRM-recognizable.
Thus, $\{c\}$ is ITTM-decidable with recursive (in fact, extremely simple) snapshots.

ITRMs refute the modified bold conjecture

Let $\alpha < \sigma$ be given. Pick $\beta < \sigma$ such that $\beta > \alpha^{+\omega}$ (the next limit of admissible ordinals after α) and β is minimal with the property that $L_\beta \models \phi$ for some $\in$-sentence ϕ.
Let c be the $<_L$-minimal real code for L_β.
By the results on ITRM-recognizability, c is ITRM-recognizable.
Thus, $\{c\}$ is ITTM-decidable with recursive (in fact, extremely simple) snapshots.

ITRMs refute the modified bold conjecture

Let $\alpha < \sigma$ be given. Pick $\beta < \sigma$ such that $\beta > \alpha^{+\omega}$ (the next limit of admissible ordinals after α) and β is minimal with the property that $L_\beta \models \phi$ for some $\in$-sentence ϕ.
Let c be the $<_L$-minimal real code for L_β.
By the results on ITRM-recognizability, c is ITRM-recognizable.
Thus, $\{c\}$ is ITTM-decidable with recursive (in fact, extremely simple) snapshots.

c has a countable decision time bound

Let P be an ITTM-program that recognizes c. We will "improve" P to one that has a countable time bound.

Consider the following routine: On input x, run P^x. As soon as P^x halts, output the result.

In parallel, run Welch's "theory machine" that successively computes codes for the L-levels $L_\gamma[x]$ with $\gamma < \lambda^x$; for each such code, search through it to determine whether L_γ contains a real number y such that $P^y \downarrow = 1$; if not, continue, otherwise, compare this real number to c, output 1 if they agree and 0 otherwise.

If $\lambda^x \geq \lambda^c$, c will be found and identified in $< \lambda^c$ many steps. If $\lambda^x < \lambda^c$, this halts in $< \lambda^c$ many steps by definition of λ^x. In any case, the halting time is bounded by λ^c.

c has a countable decision time bound

Let P be an ITTM-program that recognizes c. We will “improve” P to one that has a countable time bound.

Consider the following routine: On input x, run P^x. As soon as P^x halts, output the result.

In parallel, run Welch’s “theory machine” that successively computes codes for the L-levels $L_\gamma[x]$ with $\gamma < \lambda^x$; for each such code, search through it to determine whether L_γ contains a real number y such that $P^y \downarrow = 1$; if not, continue, otherwise, compare this real number to c, output 1 if they agree and 0 otherwise.

If $\lambda^x \geq \lambda^c$, c will be found and identified in $< \lambda^c$ many steps. If $\lambda^x < \lambda^c$, this halts in $< \lambda^c$ many steps by definition of λ^x. In any case, the halting time is bounded by λ^c.

c has a countable decision time bound

Let P be an ITTM-program that recognizes c. We will “improve” P to one that has a countable time bound.
Consider the following routine: On input x, run P^x. As soon as P^x halts, output the result.
In parallel, run Welch's “theory machine” that successively computes codes for the L-levels $L_\gamma[x]$ with $\gamma < \lambda^x$; for each such code, search through it to determine whether L_γ contains a real number y such that $P^y \downarrow = 1$; if not, continue, otherwise, compare this real number to c, output 1 if they agree and 0 otherwise.
If $\lambda^x \geq \lambda^c$, c will be found and identified in $< \lambda^c$ many steps. If $\lambda^x < \lambda^c$, this halts in $< \lambda^c$ many steps by definition of λ^x. In any case, the halting time is bounded by λ^c.

c has a countable decision time bound

Let P be an ITTM-program that recognizes c. We will "improve" P to one that has a countable time bound.

Consider the following routine: On input x, run P^x. As soon as P^x halts, output the result.

In parallel, run Welch's "theory machine" that successively computes codes for the L-levels $L_\gamma[x]$ with $\gamma < \lambda^x$; for each such code, search through it to determine whether L_γ contains a real number y such that $P^y \downarrow = 1$; if not, continue, otherwise, compare this real number to c, output 1 if they agree and 0 otherwise.

If $\lambda^x \geq \lambda^c$, c will be found and identified in $< \lambda^c$ many steps. If $\lambda^x < \lambda^c$, this halts in $< \lambda^c$ many steps by definition of λ^x. In any case, the halting time is bounded by λ^c.

c has a countable decision time bound

Let P be an ITTM-program that recognizes c. We will "improve" P to one that has a countable time bound.

Consider the following routine: On input x, run P^x. As soon as P^x halts, output the result.

In parallel, run Welch's "theory machine" that successively computes codes for the L-levels $L_\gamma[x]$ with $\gamma < \lambda^x$; for each such code, search through it to determine whether L_γ contains a real number y such that $P^y \downarrow= 1$; if not, continue, otherwise, compare this real number to c, output 1 if they agree and 0 otherwise.

If $\lambda^x \geq \lambda^c$, c will be found and identified in $< \lambda^c$ many steps. If $\lambda^x < \lambda^c$, this halts in $< \lambda^c$ many steps by definition of λ^x. In any case, the halting time is bounded by λ^c.

c has a countable decision time bound

Let P be an ITTM-program that recognizes c. We will "improve" P to one that has a countable time bound.
Consider the following routine: On input x, run P^x. As soon as P^x halts, output the result.
In parallel, run Welch's "theory machine" that successively computes codes for the L-levels $L_\gamma[x]$ with $\gamma < \lambda^x$; for each such code, search through it to determine whether L_γ contains a real number y such that $P^y \downarrow = 1$; if not, continue, otherwise, compare this real number to c, output 1 if they agree and 0 otherwise.
If $\lambda^x \geq \lambda^c$, c will be found and identified in $< \lambda^c$ many steps. If $\lambda^x < \lambda^c$, this halts in $< \lambda^c$ many steps by definition of λ^x. In any case, the halting time is bounded by λ^c.

c has a countable decision time bound

Let P be an ITTM-program that recognizes c. We will "improve" P to one that has a countable time bound.
Consider the following routine: On input x, run P^x. As soon as P^x halts, output the result.
In parallel, run Welch's "theory machine" that successively computes codes for the L-levels $L_\gamma[x]$ with $\gamma < \lambda^x$; for each such code, search through it to determine whether L_γ contains a real number y such that $P^y \downarrow = 1$; if not, continue, otherwise, compare this real number to c, output 1 if they agree and 0 otherwise.
If $\lambda^x \geq \lambda^c$, c will be found and identified in $< \lambda^c$ many steps. If $\lambda^x < \lambda^c$, this halts in $< \lambda^c$ many steps by definition of λ^x. In any case, the halting time is bounded by λ^c.

c has a countable decision time bound

Let P be an ITTM-program that recognizes c. We will “improve” P to one that has a countable time bound.
Consider the following routine: On input x, run P^x. As soon as P^x halts, output the result.
In parallel, run Welch’s “theory machine” that successively computes codes for the L-levels $L_\gamma[x]$ with $\gamma < \lambda^x$; for each such code, search through it to determine whether L_γ contains a real number y such that $P^y \downarrow= 1$; if not, continue, otherwise, compare this real number to c, output 1 if they agree and 0 otherwise.
If $\lambda^x \geq \lambda^c$, c will be found and identified in $< \lambda^c$ many steps. If $\lambda^x < \lambda^c$, this halts in $< \lambda^c$ many steps by definition of λ^x. In any case, the halting time is bounded by λ^c.

Deciding $\{c\}$ takes long

Assume for a contradiction that $\{c\}$ is ITTM-decidable with constant time bound α, say by the program P.

Then the sentence "There is a real number x such that $P^x \downarrow = 1$" is Σ_1 and true in $V_{\alpha^{+\omega}}$.

By a variant of Shoenfield absoluteness, due to Jensen and Karp, if γ is a limit of admissible ordinals, then a Σ_1-statement that holds in V_γ holds in L_γ.

It follows that $c \in L_{\alpha^{+\omega}}$. However, this implies that $L_\beta \in L_{\alpha^{+\omega}}$, while $\beta > \alpha^{+\omega}$, a contradiction.

Deciding $\{c\}$ takes long

Assume for a contradiction that $\{c\}$ is ITTM-decidable with constant time bound α, say by the program P.
Then the sentence "There is a real number x such that $P^x\downarrow = 1$" is Σ_1 and true in $V_{\alpha^{+\omega}}$.
By a variant of Shoenfield absoluteness, due to Jensen and Karp, if γ is a limit of admissible ordinals, then a Σ_1-statement that holds in V_γ holds in L_γ.
It follows that $c \in L_{\alpha^{+\omega}}$. However, this implies that $L_\beta \in L_{\alpha^{+\omega}}$, while $\beta > \alpha^{+\omega}$, a contradiction.

Deciding $\{c\}$ takes long

Assume for a contradiction that $\{c\}$ is ITTM-decidable with constant time bound α, say by the program P.
Then the sentence “There is a real number x such that $P^x \downarrow = 1$” is Σ_1 and true in $V_{\alpha^{+\omega}}$.
By a variant of Shoenfield absoluteness, due to Jensen and Karp, if γ is a limit of admissible ordinals, then a Σ_1-statement that holds in V_γ holds in L_γ.
It follows that $c \in L_{\alpha^{+\omega}}$. However, this implies that $L_\beta \in L_{\alpha^{+\omega}}$, while $\beta > \alpha^{+\omega}$, a contradiction.

Deciding $\{c\}$ takes long

Assume for a contradiction that $\{c\}$ is ITTM-decidable with constant time bound α, say by the program P.
Then the sentence "There is a real number x such that $P^x \downarrow = 1$" is Σ_1 and true in $V_{\alpha^{+\omega}}$.
By a variant of Shoenfield absoluteness, due to Jensen and Karp, if γ is a limit of admissible ordinals, then a Σ_1-statement that holds in V_γ holds in L_γ.
It follows that $c \in L_{\alpha^{+\omega}}$. However, this implies that $L_\beta \in L_{\alpha^{+\omega}}$, while $\beta > \alpha^{+\omega}$, a contradiction.

Deciding $\{c\}$ takes long

Assume for a contradiction that $\{c\}$ is ITTM-decidable with constant time bound α, say by the program P.
Then the sentence "There is a real number x such that $P^x\downarrow = 1$" is Σ_1 and true in $V_{\alpha^{+\omega}}$.
By a variant of Shoenfield absoluteness, due to Jensen and Karp, if γ is a limit of admissible ordinals, then a Σ_1-statement that holds in V_γ holds in L_γ.
It follows that $c \in L_{\alpha^{+\omega}}$. However, this implies that $L_\beta \in L_{\alpha^{+\omega}}$, while $\beta > \alpha^{+\omega}$, a contradiction.

The last resort

We now know that, for any $\alpha < \sigma$, there is a set (in fact, a singleton set) of real numbers that is ITTM-decidable with extremely simple snapshots and a uniform time bound, but not with uniform time bound α.

It is still possible that there are $\alpha \geq \sigma$ such that some sets are ITTM-decidable with uniform time bound $> \alpha$, but not with recursive snapshots.

This would be ruled out if uniform decision time bounds for ITTMs were always $< \sigma$.
Which turns out to be true.

The last resort

We now know that, for any $\alpha < \sigma$, there is a set (in fact, a singleton set) of real numbers that is ITTM-decidable with extremely simple snapshots and a uniform time bound, but not with uniform time bound α.

It is still possible that there are $\alpha \geq \sigma$ such that some sets are ITTM-decidable with uniform time bound $> \alpha$, but not with recursive snapshots.

This would be ruled out if uniform decision time bounds for ITTMs were always $< \sigma$.
Which turns out to be true.

The last resort

We now know that, for any $\alpha < \sigma$, there is a set (in fact, a singleton set) of real numbers that is ITTM-decidable with extremely simple snapshots and a uniform time bound, but not with uniform time bound α.

It is still possible that there are $\alpha \geq \sigma$ such that some sets are ITTM-decidable with uniform time bound $> \alpha$, but not with recursive snapshots.

This would be ruled out if uniform decision time bounds for ITTMs were always $< \sigma$.
Which turns out to be true.

The last resort

We now know that, for any $\alpha < \sigma$, there is a set (in fact, a singleton set) of real numbers that is ITTM-decidable with extremely simple snapshots and a uniform time bound, but not with uniform time bound α.

It is still possible that there are $\alpha \geq \sigma$ such that some sets are ITTM-decidable with uniform time bound $> \alpha$, but not with recursive snapshots.

This would be ruled out if uniform decision time bounds for ITTMs were always $< \sigma$.
Which turns out to be true.

... is gone

Theorem: If X is a set of real numbers that is ITTM-decidable with minimal constant time bound $\alpha < \omega_1$, then $\alpha < \sigma$.

Proof: Let P be an ITTM-program that decides X with constant time bound $\alpha < \omega_1$, where α is minimal.

The the statement "There is a countable ordinal α such that, for all real numbers x, P^x halts in $< \alpha$ many steps" is Σ_2^1 and holds in V.

By Shoenfield absoluteness, it holds in L.

As $L_\sigma \prec_{\Sigma_2^1} L$, it holds in L_σ. Thus $\alpha \in L_\sigma$, so $\alpha < \sigma$.

... is gone

Theorem: If X is a set of real numbers that is ITTM-decidable with minimal constant time bound $\alpha < \omega_1$, then $\alpha < \sigma$.

Proof: Let P be an ITTM-program that decides X with constant time bound $\alpha < \omega_1$, where α is minimal.

The the statement "There is a countable ordinal α such that, for all real numbers x, P^x halts in $< \alpha$ many steps" is Σ_2^1 and holds in V.

By Shoenfield absoluteness, it holds in L.

As $L_\sigma \prec_{\Sigma_2^1} L$, it holds in L_σ. Thus $\alpha \in L_\sigma$, so $\alpha < \sigma$.

... is gone

Theorem: If X is a set of real numbers that is ITTM-decidable with minimal constant time bound $\alpha < \omega_1$, then $\alpha < \sigma$.

Proof: Let P be an ITTM-program that decides X with constant time bound $\alpha < \omega_1$, where α is minimal.

The the statement "There is a countable ordinal α such that, for all real numbers x, P^x halts in $< \alpha$ many steps" is Σ^1_2 and holds in V.

By Shoenfield absoluteness, it holds in L.

As $L_\sigma \prec_{\Sigma^1_2} L$, it holds in L_σ. Thus $\alpha \in L_\sigma$, so $\alpha < \sigma$.

... is gone

Theorem: If X is a set of real numbers that is ITTM-decidable with minimal constant time bound $\alpha < \omega_1$, then $\alpha < \sigma$.

Proof: Let P be an ITTM-program that decides X with constant time bound $\alpha < \omega_1$, where α is minimal.

The the statement "There is a countable ordinal α such that, for all real numbers x, P^x halts in $< \alpha$ many steps" is Σ^1_2 and holds in V.

By Shoenfield absoluteness, it holds in L.

As $L_\sigma \prec_{\Sigma^1_2} L$, it holds in L_σ. Thus $\alpha \in L_\sigma$, so $\alpha < \sigma$.

... is gone

Theorem: If X is a set of real numbers that is ITTM-decidable with minimal constant time bound $\alpha < \omega_1$, then $\alpha < \sigma$.

Proof: Let P be an ITTM-program that decides X with constant time bound $\alpha < \omega_1$, where α is minimal.

The the statement “There is a countable ordinal α such that, for all real numbers x, P^x halts in $< \alpha$ many steps” is Σ^1_2 and holds in V.

By Shoenfield absoluteness, it holds in L.

As $L_\sigma \prec_{\Sigma^1_2} L$, it holds in L_σ. Thus $\alpha \in L_\sigma$, so $\alpha < \sigma$.

... is gone

Theorem: If X is a set of real numbers that is ITTM-decidable with minimal constant time bound $\alpha < \omega_1$, then $\alpha < \sigma$.

Proof: Let P be an ITTM-program that decides X with constant time bound $\alpha < \omega_1$, where α is minimal.

The the statement “There is a countable ordinal α such that, for all real numbers x, P^x halts in $< \alpha$ many steps” is Σ^1_2 and holds in V.

By Shoenfield absoluteness, it holds in L.

As $L_\sigma \prec_{\Sigma^1_2} L$, it holds in L_σ. Thus $\alpha \in L_\sigma$, so $\alpha < \sigma$.

One answer leads to more questions

What is the supremum of...

- countable ITTM-decision times for sets of real numbers?
- countable ITTM-decision times for singletons?
- countable ITTM-semidecision times for sets of real numbers?
- countable ITTM-semidecision times for singletons?
- countable ITTM-co-semidecision times for singletons? (For sets of reals, this is of course the same as for ITTM-semidecision times.)

One answer leads to more questions

What is the supremum of...

- countable ITTM-decision times for sets of real numbers?
- countable ITTM-decision times for singletons?
- countable ITTM-semidecision times for sets of real numbers?
- countable ITTM-semidecision times for singletons?
- countable ITTM-co-semidecision times for singletons? (For sets of reals, this is of course the same as for ITTM-semidecision times.)

One answer leads to more questions

What is the supremum of...

- countable ITTM-decision times for sets of real numbers?
- countable ITTM-decision times for singletons?
- countable ITTM-semidecision times for sets of real numbers?
- countable ITTM-semidecision times for singletons?
- countable ITTM-co-semidecision times for singletons? (For sets of reals, this is of course the same as for ITTM-semidecision times.)

One answer leads to more questions

What is the supremum of...

- countable ITTM-decision times for sets of real numbers?
- countable ITTM-decision times for singletons?
- countable ITTM-semidecision times for sets of real numbers?
- countable ITTM-semidecision times for singletons?
- countable ITTM-co-semidecision times for singletons? (For sets of reals, this is of course the same as for ITTM-semidecision times.)

One answer leads to more questions

What is the supremum of...

- countable ITTM-decision times for sets of real numbers?
- countable ITTM-decision times for singletons?
- countable ITTM-semidecision times for sets of real numbers?
- countable ITTM-semidecision times for singletons?
- countable ITTM-co-semidecision times for singletons? (For sets of reals, this is of course the same as for ITTM-semidecision times.)

One answer leads to more questions

What is the supremum of...

- countable ITTM-decision times for sets of real numbers?
- countable ITTM-decision times for singletons?
- countable ITTM-semidecision times for sets of real numbers?
- countable ITTM-semidecision times for singletons?
- countable ITTM-co-semidecision times for singletons? (For sets of reals, this is of course the same as for ITTM-semidecision times.)

One answer leads to more questions

What is the supremum of...

- countable ITTM-decision times for sets of real numbers?
- countable ITTM-decision times for singletons?
- countable ITTM-semidecision times for sets of real numbers?
- countable ITTM-semidecision times for singletons?
- countable ITTM-co-semidecision times for singletons? (For sets of reals, this is of course the same as for ITTM-semidecision times.)

One answer leads to more questions

What is the supremum of...

- countable ITTM-decision times for sets of real numbers?
- countable ITTM-decision times for singletons?
- countable ITTM-semidecision times for sets of real numbers?
- countable ITTM-semidecision times for singletons?
- countable ITTM-co-semidecision times for singletons? (For sets of reals, this is of course the same as for ITTM-semidecision times.)

σ is significant

- The supremum of countable ITTM-decision times for sets of real numbers is σ.
- The supremum of ITTM-decision times for singletons is σ. Every decidable singleton has a countable time bound.
- The supremum of countable ITTM-semidecision times for singletons is σ.
- The supremum of countable ITTM-co-semidecision times for singletons is σ. (In particular, they exist.)

The first two statements are already proved by the above.

σ is significant

- The supremum of countable ITTM-decision times for sets of real numbers is σ.
- The supremum of ITTM-decision times for singletons is σ.
  Every decidable singleton has a countable time bound.
- The supremum of countable ITTM-semidecision times for singletons is σ.
- The supremum of countable ITTM-co-semidecision times for singletons is σ. (In particular, they exist.)

The first two statements are already proved by the above.

σ is significant

- The supremum of countable ITTM-decision times for sets of real numbers is σ.
- The supremum of ITTM-decision times for singletons is σ. Every decidable singleton has a countable time bound.
- The supremum of countable ITTM-semidecision times for singletons is σ.
- The supremum of countable ITTM-co-semidecision times for singletons is σ. (In particular, they exist.)

The first two statements are already proved by the above.

σ is significant

- The supremum of countable ITTM-decision times for sets of real numbers is σ.
- The supremum of ITTM-decision times for singletons is σ. Every decidable singleton has a countable time bound.
- The supremum of countable ITTM-semidecision times for singletons is σ.
- The supremum of countable ITTM-co-semidecision times for singletons is σ. (In particular, they exist.)

The first two statements are already proved by the above.

σ is significant

- The supremum of countable ITTM-decision times for sets of real numbers is σ.
- The supremum of ITTM-decision times for singletons is σ. Every decidable singleton has a countable time bound.
- The supremum of countable ITTM-semidecision times for singletons is σ.
- The supremum of countable ITTM-co-semidecision times for singletons is σ. (In particular, they exist.)

The first two statements are already proved by the above.

σ is significant

- The supremum of countable ITTM-decision times for sets of real numbers is σ.
- The supremum of ITTM-decision times for singletons is σ. Every decidable singleton has a countable time bound.
- The supremum of countable ITTM-semidecision times for singletons is σ.
- The supremum of countable ITTM-co-semidecision times for singletons is σ. (In particular, they exist.)

The first two statements are already proved by the above.

σ is significant

- The supremum of countable ITTM-decision times for sets of real numbers is σ.
- The supremum of ITTM-decision times for singletons is σ. Every decidable singleton has a countable time bound.
- The supremum of countable ITTM-semidecision times for singletons is σ.
- The supremum of countable ITTM-co-semidecision times for singletons is σ. (In particular, they exist.)

The first two statements are already proved by the above.

But what about semidecidability?

Let us denote by σ^s the supremum of countable bounds on ITTM-semidecision times.

Soon after we started investigating this, we observed that $\sigma^s > \sigma$ and in fact that, if ω_1^L is countable, we can even have $\sigma^s > \omega_1^L$!

The problem here is that “If P^x halts, it does so in $< \alpha$ stepsl” is more complicated than “P^x halts in $< \alpha$ many steps”. It is Σ_2 rather than Σ_1.

Thus let us, in analogy with σ, define τ to be the supremum of the $\Sigma_2^{L_{\omega_1^V}}$-definable ordinals. Equivalently, τ is the supremum of the $\Pi_1^{L_{\omega_1}}$-definable ordinals.

But what about semidecidability?

Let us denote by σ^s the supremum of countable bounds on ITTM-semidecision times.

Soon after we started investigating this, we observed that $\sigma^s > \sigma$ and in fact that, if ω_1^L is countable, we can even have $\sigma^s > \omega_1^L$!

The problem here is that "If P^x halts, it does so in $< \alpha$ stepsl" is more complicated than "P^x halts in $< \alpha$ many steps". It is Σ_2 rather than Σ_1.

Thus let us, in analogy with σ, define τ to be the supremum of the $\Sigma_2^{L_{\omega_1^V}}$-definable ordinals. Equivalently, τ is the supremum of the $\Pi_1^{L_{\omega_1}}$-definable ordinals.

But what about semidecidability?

Let us denote by σ^s the supremum of countable bounds on ITTM-semidecision times.

Soon after we started investigating this, we observed that $\sigma^s > \sigma$ and in fact that, if ω_1^L is countable, we can even have $\sigma^s > \omega_1^L$!

The problem here is that "If P^x halts, it does so in $< \alpha$ stepsl" is more complicated than "P^x halts in $< \alpha$ many steps". It is Σ_2 rather than Σ_1.

Thus let us, in analogy with σ, define τ to be the supremum of the $\Sigma_2^{L_{\omega_1^V}}$-definable ordinals. Equivalently, τ is the supremum of the $\Pi_1^{L_{\omega_1}}$-definable ordinals.

But what about semidecidability?

Let us denote by σ^s the supremum of countable bounds on ITTM-semidecision times.

Soon after we started investigating this, we observed that $\sigma^s > \sigma$ and in fact that, if ω_1^L is countable, we can even have $\sigma^s > \omega_1^L$!

The problem here is that "If P^x halts, it does so in $< \alpha$ stepsl" is more complicated than "P^x halts in $< \alpha$ many steps". It is Σ_2 rather than Σ_1.

Thus let us, in analogy with σ, define τ to be the supremum of the $\Sigma_2^{L_{\omega_1^V}}$-definable ordinals. Equivalently, τ is the supremum of the $\Pi_1^{L_{\omega_1}}$-definable ordinals.

But what about semidecidability?

Let us denote by σ^s the supremum of countable bounds on ITTM-semidecision times.

Soon after we started investigating this, we observed that $\sigma^s > \sigma$ and in fact that, if ω_1^L is countable, we can even have $\sigma^s > \omega_1^L$!

The problem here is that "If P^x halts, it does so in $< \alpha$ stepsl" is more complicated than "P^x halts in $< \alpha$ many steps". It is Σ_2 rather than Σ_1.

Thus let us, in analogy with σ, define τ to be the supremum of the $\Sigma_2^{L_{\omega_1^V}}$-definable ordinals. Equivalently, τ is the supremum of the $\Pi_1^{L_{\omega_1}}$-definable ordinals.

But what about semidecidability?

Let us denote by σ^s the supremum of countable bounds on ITTM-semidecision times.

Soon after we started investigating this, we observed that $\sigma^s > \sigma$ and in fact that, if ω_1^L is countable, we can even have $\sigma^s > \omega_1^L$!

The problem here is that "If P^x halts, it does so in $< \alpha$ stepsl" is more complicated than "P^x halts in $< \alpha$ many steps". It is Σ_2 rather than Σ_1.

Thus let us, in analogy with σ, define τ to be the supremum of the $\Sigma_2^{L_{\omega_1^V}}$-definable ordinals. Equivalently, τ is the supremum of the $\Pi_1^{L_{\omega_1}}$-definable ordinals.

But what about semidecidability?

Let us denote by σ^s the supremum of countable bounds on ITTM-semidecision times.

Soon after we started investigating this, we observed that $\sigma^s > \sigma$ and in fact that, if ω_1^L is countable, we can even have $\sigma^s > \omega_1^L$!

The problem here is that "If P^x halts, it does so in $< \alpha$ stepsl" is more complicated than "P^x halts in $< \alpha$ many steps". It is Σ_2 rather than Σ_1.

Thus let us, in analogy with σ, define τ to be the supremum of the $\Sigma_2^{L_{\omega_1^V}}$-definable ordinals. Equivalently, τ is the supremum of the $\Pi_1^{L_{\omega_1}}$-definable ordinals.

Theorem: We have $\sigma^s = \tau$, i.e., the supremum of ITTM-semidecision times for sets of real numbers is equal to the supremum of the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.

Theorem: We have $\sigma^s = \tau$, i.e., the supremum of ITTM-semidecision times for sets of real numbers is equal to the supremum of the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.

Proof (Sketch):

(i) “There is $\alpha < \omega_1$ such that, if P^x halts, then it halts in $< \alpha$ many steps” is Σ^1_2; thus, if it holds, it holds in $L_\tau^{L_{\omega_1}}$ by definition of τ.

(ii) It remains to show that the semidecision times are cofinal in τ. Pick $\nu < \tau$; wlog, ν is $\Pi_1^{L_{\omega_1}}$-definable, say by the formula ϕ. Let A be the set of real numbers x such that:

- x codes an ordinal γ.
- $L_\gamma \models \exists\beta\phi(\beta)$; let ξ be the minimal witness.
- L_γ is minimal such that some $\in$-formula with parameters from $\xi + 1$ becomes true.

A is Π^1_1 and thus ITTM-(semi)decidable.

Proof (Sketch):

(i) “There is $\alpha < \omega_1$ such that, if P^x halts, then it halts in $< \alpha$ many steps” is Σ^1_2; thus, if it holds, it holds in $L_\tau^{L_{\omega_1}}$ by definition of τ.

(ii) It remains to show that the semidecision times are cofinal in τ. Pick $\nu < \tau$; wlog, ν is $\Pi_1^{L_{\omega_1}}$-definable, say by the formula ϕ. Let A be the set of real numbers x such that:

- x codes an ordinal γ.
- $L_\gamma \models \exists\beta\phi(\beta)$; let ξ be the minimal witness.
- L_γ is minimal such that some $\in$-formula with parameters from $\xi + 1$ becomes true.

A is Π^1_1 and thus ITTM-(semi)decidable.

Proof (Sketch):

(i) "There is $\alpha < \omega_1$ such that, if P^x halts, then it halts in $< \alpha$ many steps" is Σ^1_2; thus, if it holds, it holds in $L_\tau^{L_{\omega_1}}$ by definition of τ.

(ii) It remains to show that the semidecision times are cofinal in τ. Pick $\nu < \tau$; wlog, ν is $\Pi_1^{L_{\omega_1}}$-definable, say by the formula ϕ. Let A be the set of real numbers x such that:

- x codes an ordinal γ.
- $L_\gamma \models \exists \beta \phi(\beta)$; let ξ be the minimal witness.
- L_γ is minimal such that some $\in$-formula with parameters from $\xi + 1$ becomes true.

A is Π^1_1 and thus ITTM-(semi)decidable.

Proof (Sketch):

(i) "There is $\alpha < \omega_1$ such that, if P^x halts, then it halts in $< \alpha$ many steps" is Σ^1_2; thus, if it holds, it holds in $L_\tau^{L_{\omega_1}}$ by definition of τ.

(ii) It remains to show that the semidecision times are cofinal in τ. Pick $\nu < \tau$; wlog, ν is $\Pi_1^{L_{\omega_1}}$-definable, say by the formula ϕ. Let A be the set of real numbers x such that:

- x codes an ordinal γ.
- $L_\gamma \models \exists\beta\phi(\beta)$; let ξ be the minimal witness.
- L_γ is minimal such that some $\in$-formula with parameters from $\xi + 1$ becomes true.

A is Π^1_1 and thus ITTM-(semi)decidable.

Proof (Sketch):

(i) "There is $\alpha < \omega_1$ such that, if P^x halts, then it halts in $< \alpha$ many steps" is Σ^1_2; thus, if it holds, it holds in $L_\tau^{L_{\omega_1}}$ by definition of τ.

(ii) It remains to show that the semidecision times are cofinal in τ. Pick $\nu < \tau$; wlog, ν is $\Pi_1^{L_{\omega_1}}$-definable, say by the formula ϕ. Let A be the set of real numbers x such that:

- x codes an ordinal γ.
- $L_\gamma \models \exists\beta\phi(\beta)$; let ξ be the minimal witness.
- L_γ is minimal such that some $\in$-formula with parameters from $\xi + 1$ becomes true.

A is Π^1_1 and thus ITTM-(semi)decidable.

Proof (Sketch):

(i) "There is $\alpha < \omega_1$ such that, if P^x halts, then it halts in $< \alpha$ many steps" is Σ^1_2; thus, if it holds, it holds in $L_\tau^{L_{\omega_1}}$ by definition of τ.

(ii) It remains to show that the semidecision times are cofinal in τ. Pick $\nu < \tau$; wlog, ν is $\Pi_1^{L_{\omega_1}}$-definable, say by the formula ϕ. Let A be the set of real numbers x such that:

- x codes an ordinal γ.
- $L_\gamma \models \exists \beta \phi(\beta)$; let ξ be the minimal witness.
- L_γ is minimal such that some $\in$-formula with parameters from $\xi + 1$ becomes true.

A is Π^1_1 and thus ITTM-(semi)decidable.

Proof (Sketch):

(i) "There is $\alpha < \omega_1$ such that, if P^x halts, then it halts in $< \alpha$ many steps" is Σ^1_2; thus, if it holds, it holds in $L^{L_{\omega_1}}_{\tau}$ by definition of τ.

(ii) It remains to show that the semidecision times are cofinal in τ. Pick $\nu < \tau$; wlog, ν is $\Pi^{L_{\omega_1}}_1$-definable, say by the formula ϕ. Let A be the set of real numbers x such that:

- x codes an ordinal γ.
- $L_\gamma \models \exists\beta\phi(\beta)$; let ξ be the minimal witness.
- L_γ is minimal such that some $\in$-formula with parameters from $\xi + 1$ becomes true.

A is Π^1_1 and thus ITTM-(semi)decidable.

Proof (Sketch):

(i) "There is $\alpha < \omega_1$ such that, if P^x halts, then it halts in $< \alpha$ many steps" is Σ^1_2; thus, if it holds, it holds in $L_\tau^{L_{\omega_1}}$ by definition of τ.

(ii) It remains to show that the semidecision times are cofinal in τ. Pick $\nu < \tau$; wlog, ν is $\Pi_1^{L_{\omega_1}}$-definable, say by the formula ϕ. Let A be the set of real numbers x such that:

- x codes an ordinal γ.
- $L_\gamma \models \exists\beta\phi(\beta)$; let ξ be the minimal witness.
- L_γ is minimal such that some $\in$-formula with parameters from $\xi + 1$ becomes true.

A is Π^1_1 and thus ITTM-(semi)decidable.

Suppose for a contradiction that A is ITTM-semidecidable in time $\gamma < \sigma_\nu$.

Pick f and g mutually generic over L_{σ_ν} for the forcing that makes γ countable. Let x_g, x_f be real numbers that code g and f, respectively.

Then A is Σ^1_1 in both x_g and x_f. Since A is well-ordered by the $<$-relation on the coded elements, we have $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g]$ and also $A \subseteq L_{\omega_1^{\mathrm{CK},x_f}}[x_f]$.

Thus $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g] \cap L_{\omega_1^{\mathrm{CK},x_f}}[x_f] = L_{\omega_1^{\mathrm{CK},x_g}}$.

Since σ_ν is a limit of admissible ordinals and forcing over L_{σ_ν} preserves their admissibility, we have $\omega_1^{\mathrm{CK},x_g} < \sigma_\nu$.
On the other hand, A can be seen to be unbounded in L_{σ_ν}, a contradiction.

Suppose for a contradiction that A is ITTM-semidecidable in time $\gamma < \sigma_\nu$.

Pick f and g mutually generic over L_{σ_ν} for the forcing that makes γ countable. Let x_g, x_f be real numbers that code g and f, respectively.

Then A is Σ^1_1 in both x_g and x_f. Since A is well-ordered by the $<$-relation on the coded elements, we have $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g]$ and also $A \subseteq L_{\omega_1^{\mathrm{CK},x_f}}[x_f]$.

Thus $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g] \cap L_{\omega_1^{\mathrm{CK},x_f}}[x_f] = L_{\omega_1^{\mathrm{CK},x_g}}$.

Since σ_ν is a limit of admissible ordinals and forcing over L_{σ_ν} preserves their admissibility, we have $\omega_1^{\mathrm{CK},x_g} < \sigma_\nu$.
On the other hand, A can be seen to be unbounded in L_{σ_ν}, a contradiction.

Suppose for a contradiction that A is ITTM-semidecidable in time $\gamma < \sigma_\nu$.

Pick f and g mutually generic over L_{σ_ν} for the forcing that makes γ countable. Let x_g, x_f be real numbers that code g and f, respectively.

Then A is Σ^1_1 in both x_g and x_f. Since A is well-ordered by the $<$-relation on the coded elements, we have $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g]$ and also $A \subseteq L_{\omega_1^{\mathrm{CK},x_f}}[x_f]$.

Thus $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g] \cap L_{\omega_1^{\mathrm{CK},x_f}}[x_f] = L_{\omega_1^{\mathrm{CK},x_g}}$.

Since σ_ν is a limit of admissible ordinals and forcing over L_{σ_ν} preserves their admissibility, we have $\omega_1^{\mathrm{CK},x_g} < \sigma_\nu$.
On the other hand, A can be seen to be unbounded in L_{σ_ν}, a contradiction.

Suppose for a contradiction that A is ITTM-semidecidable in time $\gamma < \sigma_\nu$.

Pick f and g mutually generic over L_{σ_ν} for the forcing that makes γ countable. Let x_g, x_f be real numbers that code g and f, respectively.

Then A is Σ^1_1 in both x_g and x_f. Since A is well-ordered by the $<$-relation on the coded elements, we have $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g]$ and also $A \subseteq L_{\omega_1^{\mathrm{CK},x_f}}[x_f]$.

Thus $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g] \cap L_{\omega_1^{\mathrm{CK},x_f}}[x_f] = L_{\omega_1^{\mathrm{CK},x_g}}$.

Since σ_ν is a limit of admissible ordinals and forcing over L_{σ_ν} preserves their admissibility, we have $\omega_1^{\mathrm{CK},x_g} < \sigma_\nu$.
On the other hand, A can be seen to be unbounded in L_{σ_ν}, a contradiction.

Suppose for a contradiction that A is ITTM-semidecidable in time $\gamma < \sigma_\nu$.

Pick f and g mutually generic over L_{σ_ν} for the forcing that makes γ countable. Let x_g, x_f be real numbers that code g and f, respectively.

Then A is Σ^1_1 in both x_g and x_f. Since A is well-ordered by the $<$-relation on the coded elements, we have $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g]$ and also $A \subseteq L_{\omega_1^{\mathrm{CK},x_f}}[x_f]$.

Thus $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g] \cap L_{\omega_1^{\mathrm{CK},x_f}}[x_f] = L_{\omega_1^{\mathrm{CK},x_g}}$.

Since σ_ν is a limit of admissible ordinals and forcing over L_{σ_ν} preserves their admissibility, we have $\omega_1^{\mathrm{CK},x_g} < \sigma_\nu$.
On the other hand, A can be seen to be unbounded in L_{σ_ν}, a contradiction.

Suppose for a contradiction that A is ITTM-semidecidable in time $\gamma < \sigma_\nu$.

Pick f and g mutually generic over L_{σ_ν} for the forcing that makes γ countable. Let x_g, x_f be real numbers that code g and f, respectively.

Then A is Σ^1_1 in both x_g and x_f. Since A is well-ordered by the $<$-relation on the coded elements, we have $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g]$ and also $A \subseteq L_{\omega_1^{\mathrm{CK},x_f}}[x_f]$.

Thus $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g] \cap L_{\omega_1^{\mathrm{CK},x_f}}[x_f] = L_{\omega_1^{\mathrm{CK},x_g}}$.

Since σ_ν is a limit of admissible ordinals and forcing over L_{σ_ν} preserves their admissibility, we have $\omega_1^{\mathrm{CK},x_g} < \sigma_\nu$.
On the other hand, A can be seen to be unbounded in L_{σ_ν}, a contradiction.

Suppose for a contradiction that A is ITTM-semidecidable in time $\gamma < \sigma_\nu$.

Pick f and g mutually generic over L_{σ_ν} for the forcing that makes γ countable. Let x_g, x_f be real numbers that code g and f, respectively.

Then A is Σ^1_1 in both x_g and x_f. Since A is well-ordered by the $<$-relation on the coded elements, we have $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g]$ and also $A \subseteq L_{\omega_1^{\mathrm{CK},x_f}}[x_f]$.

Thus $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g] \cap L_{\omega_1^{\mathrm{CK},x_f}}[x_f] = L_{\omega_1^{\mathrm{CK},x_g}}$.

Since σ_ν is a limit of admissible ordinals and forcing over L_{σ_ν} preserves their admissibility, we have $\omega_1^{\mathrm{CK},x_g} < \sigma_\nu$.

On the other hand, A can be seen to be unbounded in L_{σ_ν}, a contradiction.

Suppose for a contradiction that A is ITTM-semidecidable in time $\gamma < \sigma_\nu$.

Pick f and g mutually generic over L_{σ_ν} for the forcing that makes γ countable. Let x_g, x_f be real numbers that code g and f, respectively.

Then A is Σ^1_1 in both x_g and x_f. Since A is well-ordered by the $<$-relation on the coded elements, we have $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g]$ and also $A \subseteq L_{\omega_1^{\mathrm{CK},x_f}}[x_f]$.

Thus $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g] \cap L_{\omega_1^{\mathrm{CK},x_f}}[x_f] = L_{\omega_1^{\mathrm{CK},x_g}}$.

Since σ_ν is a limit of admissible ordinals and forcing over L_{σ_ν} preserves their admissibility, we have $\omega_1^{\mathrm{CK},x_g} < \sigma_\nu$.
On the other hand, A can be seen to be unbounded in L_{σ_ν}, a contradiction.

Suppose for a contradiction that A is ITTM-semidecidable in time $\gamma < \sigma_\nu$.

Pick f and g mutually generic over L_{σ_ν} for the forcing that makes γ countable. Let x_g, x_f be real numbers that code g and f, respectively.

Then A is Σ^1_1 in both x_g and x_f. Since A is well-ordered by the $<$-relation on the coded elements, we have $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g]$ and also $A \subseteq L_{\omega_1^{\mathrm{CK},x_f}}[x_f]$.

Thus $A \subseteq L_{\omega_1^{\mathrm{CK},x_g}}[x_g] \cap L_{\omega_1^{\mathrm{CK},x_f}}[x_f] = L_{\omega_1^{\mathrm{CK},x_g}}$.

Since σ_ν is a limit of admissible ordinals and forcing over L_{σ_ν} preserves their admissibility, we have $\omega_1^{\mathrm{CK},x_g} < \sigma_\nu$.

On the other hand, A can be seen to be unbounded in L_{σ_ν}, a contradiction.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

A bit of τ-ism

τ is equal to the supremum of...

1. the countable ITTM/ITRM/OTM-semidecision times.
2. the $\Pi_1^{L_{\omega_1}}$-definable ordinals.
3. the $\Sigma_2^{L_{\omega_1}}$-definable ordinals.
4. the minimal elements of $\Pi_1^{H_{\omega_1}}$-definable subsets of ω_1.
5. the countable lengths of Π_1^1-prewellorders on Π_1^1-sets.
6. minimal elements of nonempty Π_2^1-subsets of WO, i.e., the ordinal γ_2^1 introduced by Kechris.
7. levels of the Borel hierarchy at which Σ_1^1-sets appear (Kechris/Marker/Sami) ... and many more ordinals defined via ranks.

Moreover, if ω_1^L is countable, then $\tau > \omega_1^L$. If ω_1 is inaccessible in L (e.g., when $0^\sharp$ exists), then $\tau = \aleph_\tau^L$.

Thank you for your attention!

References

- A. Kechris, D. Marker, R. Sami. Π^1_1 Borel Sets. JSL (1989)
- B. Löwe. Space Bounds for Infinitary Computation. (CiE 2006 Proceedings)
- M. Carl. Space and Time Complexity for Infinite Time Turing Machines. JLC (2020)
- M. Carl, P. Schlicht, P. Welch. Decision Times of Infinite Computations. arXiv:2011.04942v1 (submitted, 2020)
- M. Carl, P. Schlicht, P. Welch. Countable Ranks at the First and Second Projective Level. (unpublished notes, 2020)