Well-Founded Iterations of Infinite Time Turing Machines

Robert S. Lubarsky
Florida Atlantic University

August 11, 2009

Applications

Useful for ordinal analysis

Applications

Useful for ordinal analysis
Iteration and hyper-iteration/feedback

Applications

Useful for ordinal analysis
Iteration and hyper-iteration/feedback

- Turing jump $\mapsto$ hyperarithmetic sets

Applications

Useful for ordinal analysis
Iteration and hyper-iteration/feedback

- Turing jump $\mapsto$ hyperarithmetic sets
- Inductive definitions $\mapsto$ the $\mu-$calculus

Applications

Useful for ordinal analysis
Iteration and hyper-iteration/feedback

- Turing jump $\mapsto$ hyperarithmetic sets
- Inductive definitions $\mapsto$ the $\mu-$calculus
- ITTMs $\mapsto$???

First Definitions

(Hamkins & Lewis) An *Infinite time Turing machine* is a regular Turing machine with limit stages.

First Definitions

(Hamkins & Lewis) An *Infinite time Turing machine* is a regular Turing machine with limit stages. At a limit stage:

First Definitions

(Hamkins & Lewis) An *Infinite time Turing machine* is a regular Turing machine with limit stages. At a limit stage:

- the machine is in a dedicated state

First Definitions

(Hamkins & Lewis) An *Infinite time Turing machine* is a regular Turing machine with limit stages. At a limit stage:

- the machine is in a dedicated state
- the head is on the 0^{th} cell

First Definitions

(Hamkins & Lewis) An *Infinite time Turing machine* is a regular Turing machine with limit stages. At a limit stage:

- the machine is in a dedicated state
- the head is on the 0^{th} cell
- the content of a cell is limsup of the previous contents (i.e. 0 if eventually 0, 1 if eventually 1, 1 if cofinally alternating)

Writable reals and ordinals

Definition

$R \subseteq \omega$ is **writable**

if its characteristic function is on the output tape at the end of a halting computation.

Writable reals and ordinals

Definition

$R \subseteq \omega$ is **writable**
if its characteristic function is on the output tape at the end of a halting computation.

An ordinal α is **writable**
if some real coding α (via some standard representation) is writable.

Writable reals and ordinals

Definition

$R \subseteq \omega$ is **writable**
if its characteristic function is on the output tape at the end of a halting computation.

An ordinal α is **writable**
if some real coding α (via some standard representation) is writable.

$\lambda :=$ sup $\{\alpha \mid \alpha$ is writable$\}$

Writable reals and ordinals

Definition

$R \subseteq \omega$ is **writable**
if its characteristic function is on the output tape at the end of a halting computation.

An ordinal α is **writable**
if some real coding α (via some standard representation) is writable.

$\lambda :=$ sup $\{\alpha \mid \alpha$ is writable$\}$

Proposition

$R \subseteq \omega$ is writable iff $R \in L_\lambda$.

Eventually writable reals and ordinals

Definition

$R \subseteq \omega$ is **eventually writable**
if its characteristic function is on the output tape, never to change, of a computation.

An ordinal α is **eventually writable**
if some real coding α (via some standard representation) is eventually writable.

$\zeta :=$ sup $\{\alpha \mid \alpha$ is eventually writable$\}$

Proposition

$R \subseteq \omega$ is eventually writable iff $R \in L_\zeta$.

Accidentally writable reals and ordinals

Definition

$R \subseteq \omega$ is **accidentally writable**
if its characteristic function is on the output tape at any time during a computation.

An ordinal α is **accidentally writable**
if some real coding α (via some standard representation) is accidentally writable.

$\Sigma :=$ sup $\{\alpha \mid \alpha$ is accidentally writable$\}$

Proposition

$R \subseteq \omega$ is accidentally writable iff $R \in L_\Sigma$.

Summary and conclusions

λ is the supremum of the writables.
ζ is the supremum of the eventually writables.
Σ is the supremum of the accidentally writables.
Clearly, $\lambda \leq \zeta \leq \Sigma$.

Summary and conclusions

λ is the supremum of the writables.
ζ is the supremum of the eventually writables.
Σ is the supremum of the accidentally writables.
Clearly, $\lambda \leq \zeta \leq \Sigma$.

Theorem

*(Welch) ζ is the least ordinal α such that L_α has a Σ_2-elementary extension. (ζ is the least Σ_2-**extendible** ordinal.) The ordinal of that extension is Σ. L_λ is the least Σ_1-elementary substructure of L_ζ.*

Time to iterate

Definition

$0^{\blacktriangledown} = \{(e, x) | \phi_e(x) \downarrow \}$

Time to iterate

Definition

$0^\blacktriangledown = \{(e,x) | \phi_e(x)\downarrow\}$

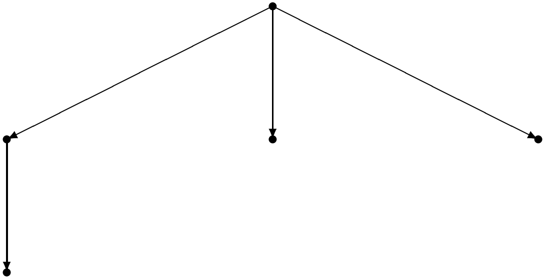
Proposition

The definitions of λ, ζ, and Σ relativize (to $\lambda^\blacktriangledown, \zeta^\blacktriangledown$, and $\Sigma^\blacktriangledown$) to computations from $0^\blacktriangledown$. Furthermore, $\zeta^\blacktriangledown$ is the least Σ_2-extendible limit of Σ_2-extendibles, the ordinal of its Σ_2 extension is $\Sigma^\blacktriangledown$, and $\lambda^\blacktriangledown$ is the ordinal of its least Σ_1-elementary substructure.

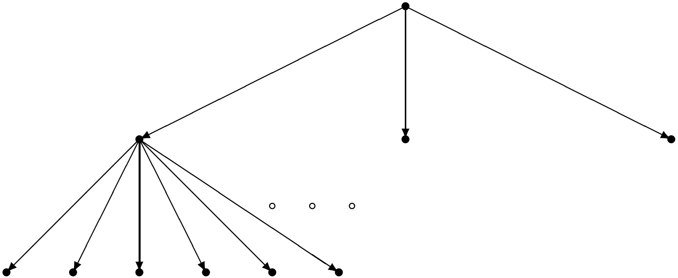
Time to iterate

ITTMs with arbitrary iteration:
A computation may ask a convergence question about another computation. This can be considered calling a sub-computation. That sub-computation might do the same. This can continue, generating a *tree of sub-computations*. Eventually, perhaps, a computation is run which calls no sub-computation. This either converges or diverges. That answer is returned to its calling computation, which then continues.

Good examples



Good examples



Bad example

Bad example

One can naturally define the course of a computation if and only if the tree of sub-computations is well-founded. How is this to be dealt with?

Option I

When the main computation makes a sub-call, the call must be made with an ordinal. When a sub-call makes a sub-call itself, that must be done with a smaller ordinal. The definitions of λ, ζ, and Σ relativize (to $\lambda^{it\blacktriangledown}, \zeta^{it\blacktriangledown}$, and $\Sigma^{it\blacktriangledown}$).

Results

Definition

β is 0- (or 1-) extendible if its Σ_2-extendible.
β is $(\alpha+1)$-extendible if its a Σ_2-extendible limit of α-extendibles.
β is κ-extendible if its a Σ_2-extendible limit of α-extendibles for each $\alpha < \kappa$.

Results

Definition

β is 0- (or 1-) extendible if its Σ_2-extendible.
β is $(\alpha+1)$-extendible if its a Σ_2-extendible limit of α-extendibles.
β is κ-extendible if its a Σ_2-extendible limit of α-extendibles for each $\alpha < \kappa$.

Proposition

$\zeta^{it\blacktriangledown}$ is the least κ which is κ-extendible, $\Sigma^{it\blacktriangledown}$ is its Σ_2 extension, and $\lambda^{it\blacktriangledown}$ its least Σ_1 substructure.

Option II

Allow all possible sub-computation calls, even if the tree of sub-computations is ill-founded, and consider only those for which the tree of sub-computations just so happens to be well-founded.

Option II

Allow all possible sub-computation calls, even if the tree of sub-computations is ill-founded, and consider only those for which the tree of sub-computations just so happens to be well-founded. So some legal computations have an undefined result. Still, among those with a defined result, some computations are halting, and some divergent computations have a stable output.

Results

BIG FACT

If a real is eventually writable in this fashion, then it's writable.

Results

BIG FACT

If a real is eventually writable in this fashion, then it's writable.

Proof.

Given e, run the computation of ϕ_e. Keep asking "if I continue running this computation until cell 0 changes, is that computation convergent or divergent?" Eventually you will get "divergent" as your answer. Then go on to cell 1, then cell 2, etc. After going through all the natural numbers, you know the real on your output tape is the eventually writable real you want. So halt. $\square$

Results

BIG FACT
If a real is eventually writable in this fashion, then it's writable.

SECOND BIG FACT
If a real is accidentally writable in this fashion, then it's writable.

Results

QUESTION
Why isn't this a contradiction? Why can't you diagonalize?

Results

QUESTION

Why isn't this a contradiction? Why can't you diagonalize?

ANSWER

You can't run a universal machine.

As soon as a machine with code for a universal machine makes an ill-founded sub-computation call, it freezes.

Prospects

Definition

R is **freezingly writable** if R appears anytime during such a computation, even if that computation later freezes.

Prospects

Definition

R is **freezingly writable** if R appears anytime during such a computation, even if that computation later freezes.

Claim In order to understand the writable reals in this context, one needs to understand the freezingly writable reals. One also needs to understand the tree of sub-computations for freezing computations.

Prospects

Notation Let Λ be the supremum of the ordinals so writable (i.e. with well-founded oracle calls).

Prospects

Notation Let Λ be the supremum of the ordinals so writable (i.e. with well-founded oracle calls).

Then in the sub-computation tree of a freezing computation either:

Prospects

Notation Let Λ be the supremum of the ordinals so writable (i.e. with well-founded oracle calls).
Then in the sub-computation tree of a freezing computation either:
a) some node has more than Λ-many children, or

Prospects

Notation Let Λ be the supremum of the ordinals so writable (i.e. with well-founded oracle calls).
Then in the sub-computation tree of a freezing computation either:
a) some node has more than Λ-many children, or
b) every level has size less than Λ, but those sizes are cofinal in Λ, or

Prospects

Notation Let Λ be the supremum of the ordinals so writable (i.e. with well-founded oracle calls).

Then in the sub-computation tree of a freezing computation either:

a) some node has more than Λ-many children, or

b) every level has size less than Λ, but those sizes are cofinal in Λ, or

c) the total number of nodes is bounded beneath Λ.

Prospects

Notation Let Λ be the supremum of the ordinals so writable (i.e. with well-founded oracle calls).

Then in the sub-computation tree of a freezing computation either:

a) some node has more than Λ-many children, or

b) every level has size less than Λ, but those sizes are cofinal in Λ, or

c) the total number of nodes is bounded beneath Λ.

Proposition

Options a) and b) are incompatible: there cannot be one tree of sub-computations with more than Λ-much splitting beneath a node and another with the splittings beneath all the nodes cofinal in Λ.

Option III

Yet another option: parallel computation:

Option III

Yet another option: parallel computation:
An oracle call may be the question “does one of these computations converge?” The computation asked about has an index e, parameter x, and free variable n.

Option III

Yet another option: parallel computation:
An oracle call may be the question "does one of these computations converge?" The computation asked about has an index e, parameter x, and free variable n. If one (natural number) value for n yields a convergent computation, the answer is "yes", even if other values yield freezing computations. The answer "yes" means some natural number yields a convergent computation, even if other numbers yield freezing. The answer "no" means all parameter values yield non-freezing computations and all are divergent.

Option III

Yet another option: parallel computation:
An oracle call may be the question "does one of these computations converge?" The computation asked about has an index e, parameter x, and free variable n. If one (natural number) value for n yields a convergent computation, the answer is "yes", even if other values yield freezing computations. The answer "yes" means some natural number yields a convergent computation, even if other numbers yield freezing. The answer "no" means all parameter values yield non-freezing computations and all are divergent. (Notice this is the same question as "does one of these computations diverge?", since divergence and convergence can be interchanged.)

Option III

Yet another option: parallel computation:
An oracle call may be the question "does one of these computations converge?" The computation asked about has an index e, parameter x, and free variable n. If one (natural number) value for n yields a convergent computation, the answer is "yes", even if other values yield freezing computations. The answer "yes" means some natural number yields a convergent computation, even if other numbers yield freezing. The answer "no" means all parameter values yield non-freezing computations and all are divergent. (Notice this is the same question as "does one of these computations diverge?", since divergence and convergence can be interchanged.)
to be continued ...

References

- Joel Hamkins and Andy Lewis, "Infinite Time Turing Machines," **The Journal of Symbolic Logic**, v. 65 (2000), p. 567-604
- Robert Lubarsky, "ITTMs with Feedback," to appear
- Philip Welch, "The Length of Infinite Time Turing Machine Computations," **The Bulletin of the London Mathematical Society**, v. 32 (2000), p. 129-136
- Philip Welch, "Eventually Infinite Time Turing Machine Degrees: Infinite Time Decidable Reals," **The Journal of Symbolic Logic**, v. 65 (2000), p. 1193-1203
- Philip Welch, "Characteristics of Discrete Transfinite Turing Machine Models: Halting Times, Stabilization Times, and Normal Form Theorems," **Theoretical Computer Science**, v. 410 (2009), p. 426-442