Randomness and Computability 3

Rod Downey
Victoria University
Wellington
New Zealand

LOWNESS

- I would like to discuss the remarkable story of lowness.

LOWNESS

- I would like to discuss the remarkable story of lowness.
- I will try to explain the decanter method, which is relatively poorly understood.

K-TRIVIALITY

- Chaitin proved that a real A is computable iff for all n, $C(A \restriction n) \leq^+ \log n$, iff $C(A \restriction n) \leq^+ C(n)$.
- This is proven using the fact that a Π^0_1 class with a finite number of paths has computable paths, combined with the Counting Theorem $\{\sigma : C(\sigma) \leq C(n) + d \wedge |\sigma| = n\} \leq A2^d$. (The Loveland Technique)

K-TRIVIALITY

- Chaitin proved that a real A is computable iff for all n, $C(A \upharpoonright n) \leq^+ \log n$, iff $C(A \upharpoonright n) \leq^+ C(n)$.
- This is proven using the fact that a Π^0_1 class with a finite number of paths has computable paths, combined with the Counting Theorem $\{\sigma : C(\sigma) \leq C(n) + d \wedge |\sigma| = n\} \leq A2^d$. (The Loveland Technique)
- What is $K(A \upharpoonright n) \leq^+ K(n)$ for all n? We call such reals K-trivial. Does A K-trivial imply A computable?
- Write $A \in KT(d)$ iff for all n, $K(A \upharpoonright n) \leq K(n) + d$.

The argument fails

- It is still true that $\{\sigma : K(\sigma) \leq K(|\sigma|) + d\}$ is $O(2^d)$, so it would appear that we could run the Π^0_1 class argument used for C. But no...
- The problem is that we don't know $K(n)$ in any computable interval, therefore the tree of K-trivials we would construct would be a Π^0_1 class relative to $\emptyset'$.

Theorem (Chaitin, Zambella)

There are only $O(2^d)$ members of $KT(d)$. They are all Δ^0_2.

Theorem (Solovay)

There are noncomputable K-trivial reals.

Theorem (Zambella)

Such reals can be c.e. sets.

THE TAILWEIGHT GAME

- The following argument due to Downey, Hirschfeldt, Nies and Stephan and independently Kummer is becomming reasonably well known to the experts, but perhaps not outside the area.
- The method could be described as the tailsum game.
- A complicated way to "prove" that the set $B = \{0^n : n \in \omega\}$ has the same complexity as $\omega = \{1^n : n \in \omega\}$.
- Opponent enumerates the universal U describing n with descriptions. We build M, a KC set o that if opponent plays (k, n) (that is 1^n has a description of length k), then we enumerate $(k, 0^n)$ into M (or indeed $(k+1, 0^n)$ would be okay too.

- In the above we play into M no more than U plays into its domain. Thus M is a KC set. The overall weight or measure of the domain of M is Ω, or $\frac{1}{2}\Omega$ if we use the "$+1$" option.
- Now imagine we are building B but now we want to make B noncomputable. Now there is asymmetry.

- In the above we play into M no more than U plays into its domain. Thus M is a KC set. The overall weight or measure of the domain of M is Ω, or $\frac{1}{2}\Omega$ if we use the "$+1$" option.
- Now imagine we are building B but now we want to make B noncomputable. Now there is asymmetry.
- To make B noncomputable, at some stage we must make $B_{s+1}(x) = 1$, where $B_s(x) = 0$.

- In the above we play into M no more than U plays into its domain. Thus M is a KC set. The overall weight or measure of the domain of M is Ω, or $\frac{1}{2}\Omega$ if we use the "$+1$" option.
- Now imagine we are building B but now we want to make B noncomputable. Now there is asymmetry.
- To make B noncomputable, at some stage we must make $B_{s+1}(x) = 1$, where $B_s(x) = 0$.
- Thus we must issue new descriptions for all of the tail: $B_{s+1} \upharpoonright n$ for $x \leq n \leq s$.
- However, A has not changed, so this requires new quanta we can't charge to U.

- In the above we play into M no more than U plays into its domain. Thus M is a KC set. The overall weight or measure of the domain of M is Ω, or $\frac{1}{2}\Omega$ if we use the "$+1$" option.
- Now imagine we are building B but now we want to make B noncomputable. Now there is asymmetry.
- To make B noncomputable, at some stage we must make $B_{s+1}(x) = 1$, where $B_s(x) = 0$.
- Thus we must issue new descriptions for all of the tail: $B_{s+1} \upharpoonright n$ for $x \leq n \leq s$.
- However, A has not changed, so this requires new quanta we can't charge to U.
- The cost is the weight of the tail, the tailsum:

$$\sum_{x \leq n \leq s} 2^{-K_s(n)}.$$

- The point is that if we can limit this cost to, say, $\frac{1}{2}$ the extra cost would be acceptable.
- Since cost equals chargable cost ($\frac{1}{2}\Omega$) plus extra cost ($\frac{1}{2}$).

- The point is that if we can limit this cost to, say, $\frac{1}{2}$ the extra cost would be acceptable.
- Since cost equals chargable cost ($\frac{1}{2}\Omega$) plus extra cost ($\frac{1}{2}$).
- $$A = \{\langle e, n\rangle : \exists s\,(W_{e,s} \cap A_s = \emptyset \wedge \langle e, n\rangle \in W_{e,s}$$

  and
  $$\wedge \sum_{\langle e,n\rangle \leq j \leq s} 2^{-K(j)[s]} < 2^{-(e+2)})\}.$$

THE DECANTER METHOD

- K-trivials form a remarkable class as we will see.
- First they solve Post's problem.
- Theorem: (DHNS) If A is K-trivial then $A <_T \emptyset'$.
- More later.
- The proof below is the version discovered by Nies.
- The proof below runs the same way whether A is Δ_2^0 or computably enumerable. We only need the relevant approximation being $A = \cup_s A_s$ or $A = \lim_s A_s$.

wtt-INCOMPLETENESS

- Tool: amplification.

wtt-INCOMPLETENESS

- Tool: amplification.
- A is in $KT(b)$, and we are building a machine M whose coding constant in U is b.
- Meaning: we describe n by some KC-axiom $\langle p, n \rangle$ i.e. we M-describe n by something of length p, then in U we describe n by something of length $p + d$ and hence the opponent at some stage s must eventually give a description of $A_s \upharpoonright n$ of length $p + b + d$.

wtt-INCOMPLETENESS

- Tool: amplification.
- A is in $KT(b)$, and we are building a machine M whose coding constant in U is b.
- Meaning: we describe n by some KC-axiom $\langle p, n\rangle$ i.e. we M-describe n by something of length p, then in U we describe n by something of length $p + d$ and hence the opponent at some stage s must eventually give a description of $A_s \upharpoonright n$ of length $p + b + d$.
- Note: the opponent has to play less quanta than we do for the same effect.

- Suppose that A is *wtt*-complete computing B (a set we build) with a known reduction $\Gamma^A = B$, and use $\gamma(x)$.
- We are trying to claim that A is not K-trivial.

- Suppose that A is *wtt*-complete computing B (a set we build) with a known reduction $\Gamma^A = B$, and use $\gamma(x)$.
- We are trying to claim that A is not K-trivial.
- We force U to issue too many descriptions of A, by using up all of its quanta.
- The first idea is to make the opponent play many times on the same length and hence amount of quanta.

- Pick $k = 2^{b+d+1}$ many followers $m_k < dots < m_1$ targeted for B and wait for a stage where $\ell(s) > m_1$, $\ell(s)$ denoting the length of agreement of $\Gamma^A = B[s]$.
- Now load an M-description of some fresh, unseen $n > \gamma(m_1)$ (and hence bigger than $\gamma(m_i)$ for all i) of size 1, enumerating an axiom $\langle 1, n \rangle$.
- Wait for the opponent to U-decribe $A_s \upharpoonright n$ with complexity $\leq 2^{-b+d}$.

- When $\ell(s) > m_1$, put m_1 into $B_{s+1} - B_s$.

- When $\ell(s) > m_1$, put m_1 into $B_{s+1} - B_s$.
- After $A_{s_1} \upharpoonright n \neq A_s \upharpoonright n$ (as $n > \gamma(m_1)$), and $\ell(s_1) > m_1$ and $K_s(A_{s_1} \upharpoonright n) \leq d + b$, again, repeat by putting m_2 into B_{s_1+1}.

- When $\ell(s) > m_1$, put m_1 into $B_{s+1} - B_s$.
- After $A_{s_1} \upharpoonright n \neq A_s \upharpoonright n$ (as $n > \gamma(m_1)$), and $\ell(s_1) > m_1$ and $K_s(A_{s_1} \upharpoonright n) \leq d + b$, again, repeat by putting m_2 into B_{s_1+1}.
- This cannot return 2^{b+d+1} many times as each time U has to issue new $A_t \upharpoonright n$ descriptions of size $\leq 2^{b+d}$.

Impossible constants

- Now, Γ is a Turing reduction; NB $\gamma(m_i, s)$ not $\gamma(m_i)$.
- Now when we play the M-description of n, the opponent can

Impossible constants

- Now, Γ is a Turing reduction; NB $\gamma(m_i, s)$ not $\gamma(m_i)$.
- Now when we play the M-description of n, the opponent can
- move the use $\gamma(m_1, s)$ (or $\gamma(m_k, s)$ even) to some value bigger than n

IMPOSSIBLE CONSTANTS

- Now, Γ is a Turing reduction; NB $\gamma(m_i, s)$ not $\gamma(m_i)$.
- Now when we play the M-description of n, the opponent can
- move the use $\gamma(m_1, s)$ (or $\gamma(m_k, s)$ even) to some value bigger than n
- before he decides to match our description of n.

Impossible constants

- Now, Γ is a Turing reduction; NB $\gamma(m_i, s)$ not $\gamma(m_i)$.
- Now when we play the M-description of n, the opponent can
- move the use $\gamma(m_1, s)$ (or $\gamma(m_k, s)$ even) to some value bigger than n
- before he decides to match our description of n.
- This costs him little.

- We realize that it is pretty dumb of us to try to describe n in one hit.
- All that really matters is that we load lots of quanta beyond some point were it is measured many times.
- Note wven for *wtt*, we certainly could have used many n's beyond $\gamma(m_1)$ loading each with, say, 2^{-e} for some small e, and only attacking once we have amassed the requisite amount beyond $\gamma(m_1)$.

- Impossible assumption: Turing reduction $\Gamma^A = B$ and the overheads of the coding and Recursion Theorem result in a constant of 0 for the coding, and the constant of triviality is 0.
- $\Gamma^A = B[s]$

- Impossible assumption: Turing reduction $\Gamma^A = B$ and the overheads of the coding and Recursion Theorem result in a constant of 0 for the coding, and the constant of triviality is 0.
- $\Gamma^A = B[s]$
- Remember, if we use the dumb strategy, then he will change $A_s \upharpoonright \gamma(m, s)$ moving some Γ-use before he describes $A_s \upharpoonright n$.
- Thus he only needs to describe $A_s \upharpoonright n$ once.

Dripfeeding

- We use a drip feed strategy for loading.
- Our goal if to load $\frac{7}{8}$ beyond some n (more or less) and have it counted twice, so he'd need $\frac{7}{4}$ in his dom$U$.
- It might be that whilst we are trying to load some quanta, the change use problem might happen, a certain amount of "trash", that is, axioms enumerated into M that do not cause the appropriate number of short descriptions to appear in U.

Dripfeeding

- We use a drip feed strategy for loading.
- Our goal if to load $\frac{7}{8}$ beyond some n (more or less) and have it counted twice, so he'd need $\frac{7}{4}$ in his dom$U$.
- It might be that whilst we are trying to load some quanta, the change use problem might happen, a certain amount of "trash", that is, axioms enumerated into M that do not cause the appropriate number of short descriptions to appear in U.
- We arrange things so that this trash is small enough to be negligable.

Inductive procedures

- We begin by using a procedure $P(\frac{7}{8}, \frac{1}{8})$, asking us for twice counted quanta (a “2-set”) of size $\frac{7}{8}$ but only having trash bounded by $\frac{1}{8}$.

INDUCTIVE PROCEDURES

- We begin by using a procedure $P(\frac{7}{8}, \frac{1}{8})$, asking us for twice counted quanta (a “2-set”) of size $\frac{7}{8}$ but only having trash bounded by $\frac{1}{8}$.
- $\frac{1}{8} = \sum_j 2^{-(j+4)}$.

INDUCTIVE PROCEDURES

- We begin by using a procedure $P(\frac{7}{8}, \frac{1}{8})$, asking us for twice counted quanta (a "2-set") of size $\frac{7}{8}$ but only having trash bounded by $\frac{1}{8}$.
- $\frac{1}{8} = \sum_j 2^{-(j+4)}$.
- Initially we might try loading quanta beyond the current use $\gamma(m, s_0)$ in lots of 2^{-4}.
- If we are successful in reaching our target of $\frac{7}{8}$ before A changes, then we are in the *wtt*-case and can simply change B to get the quanta counted twice.

- We load the quanta 2^{-4} on some $n_0 > \gamma(m, s_0)$.

- We load the quanta 2^{-4} on some $n_0 > \gamma(m, s_0)$.
- He has a choice:

- We load the quanta 2^{-4} on some $n_0 > \gamma(m, s_0)$.
- He has a choice:
- Move $\gamma(m, s)$ to some new $\gamma(m, s_1) > n_0$, at essentially no cost to him.

- We load the quanta 2^{-4} on some $n_0 > \gamma(m, s_0)$.
- He has a choice:
- Move $\gamma(m, s)$ to some new $\gamma(m, s_1) > n_0$, at essentially no cost to him.
- We played 2^{-4} for no gain, and would throw the 2^{-4} into the trash.

- We load the quanta 2^{-4} on some $n_0 > \gamma(m, s_0)$.
- He has a choice:
- Move $\gamma(m, s)$ to some new $\gamma(m, s_1) > n_0$, at essentially no cost to him.
- We played 2^{-4} for no gain, and would throw the 2^{-4} into the trash.
- Now we would begin to try to load anew $\frac{7}{8}$ beyond $\gamma(m, s_1)$ but this time we would use chunks of size 2^{-5}.

- We load the quanta 2^{-4} on some $n_0 > \gamma(m, s_0)$.
- He has a choice:
- Move $\gamma(m, s)$ to some new $\gamma(m, s_1) > n_0$, at essentially no cost to him.
- We played 2^{-4} for no gain, and would throw the 2^{-4} into the trash.
- Now we would begin to try to load anew $\frac{7}{8}$ beyond $\gamma(m, s_1)$ but this time we would use chunks of size 2^{-5}.
- Again if he moved immediately, then we would trash that quanta and next time use 2^{-6}.

- We load the quanta 2^{-4} on some $n_0 > \gamma(m, s_0)$.
- He has a choice:
- Move $\gamma(m, s)$ to some new $\gamma(m, s_1) > n_0$, at essentially no cost to him.
- We played 2^{-4} for no gain, and would throw the 2^{-4} into the trash.
- Now we would begin to try to load anew $\frac{7}{8}$ beyond $\gamma(m, s_1)$ but this time we would use chunks of size 2^{-5}.
- Again if he moved immediately, then we would trash that quanta and next time use 2^{-6}.
- Note: $\Gamma^A = B$ this movement can't happen forever, lest $\gamma(m, s) \to \infty$.

- On the other hand, in the first instance, perhaps we loaded 2^{-4} beyond $\gamma(m, s_0)$ and he did not move $\gamma(m, s_0)$ at that stage, but simply described $A \upharpoonright n_0$ by some description of size 4.

- On the other hand, in the first instance, perhaps we loaded 2^{-4} beyond $\gamma(m, s_0)$ and he did not move $\gamma(m, s_0)$ at that stage, but simply described $A \upharpoonright n_0$ by some description of size 4.
- At the next step, we would pick another n beyond $\gamma_(m, s_0) = \gamma(m, s_1)$ and try again to load 2^{-4}.

- On the other hand, in the first instance, perhaps we loaded 2^{-4} beyond $\gamma(m, s_0)$ and he did not move $\gamma(m, s_0)$ at that stage, but simply described $A \restriction n_0$ by some description of size 4.
- At the next step, we would pick another n beyond $\gamma_(m, s_0) = \gamma(m, s_1)$ and try again to load 2^{-4}.
- If the opponent now changes, then we lose the second 2^{-4} but he must count the first one (on n_0) twice.

- On the other hand, in the first instance, perhaps we loaded 2^{-4} beyond $\gamma(m, s_0)$ and he did not move $\gamma(m, s_0)$ at that stage, but simply described $A \upharpoonright n_0$ by some description of size 4.
- At the next step, we would pick another n beyond $\gamma_(m, s_0) = \gamma(m, s_1)$ and try again to load 2^{-4}.
- If the opponent now changes, then we lose the second 2^{-4} but he must count the first one (on n_0) twice.
- That is, whenever he actually does not move $\gamma(m, s)$ then he must match our description of the current n, and this will later be counted twice since either he moves $\gamma(m, s)$ over it (causing it to me counted twice) or we put m into B making $\gamma(m, s)$ change.

SUMMARY

Each time we try to load, he either matches us (in which case the amount will contribute to the 2-set, and we can return $2^{-\text{current } \beta}$ where β is the current number being used for the loading to the target,
or we lose β, but gain in that $\gamma(m, s)$ moves again, and we put β in the trash, but make the next $\beta = \frac{\beta}{2}$.

The less impossible case

- The key idea from the *wwt*-case where the use is fixed but the coding constants are nontrivial, is that we must make the changes beyond $\gamma(m_k)$ a $k-$set.

THE LESS IMPOSSIBLE CASE

- The key idea from the *wwt*-case where the use is fixed but the coding constants are nontrivial, is that we must make the changes beyond $\gamma(m_k)$ a $k-$set.
- We pretend that the constant of triviality is 0, but now the coding constant is 1.

The less impossible case

- The key idea from the *wwt*-case where the use is fixed but the coding constants are nontrivial, is that we must make the changes beyond $\gamma(m_k)$ a $k-$set.
- We pretend that the constant of triviality is 0, but now the coding constant is 1.
- Thus when we play 2^{-q} to describe some n , the opponent will only use $2^{-(q+1)}$.

- Emulating the wtt-case, we would be working with $k = 2^{1+1} = 4$ and would try to construct a 4-set of changes.

- Emulating the wtt-case, we would be working with $k = 2^{1+1} = 4$ and would try to construct a 4-set of changes.
- What we will do is break the task into the construction of a 2-set of a certain weight, a 3-set and a 4-set of a related weight in a coherent way.

- Emulating the wtt-case, we would be working with $k = 2^{1+1} = 4$ and would try to construct a 4-set of changes.
- What we will do is break the task into the construction of a 2-set of a certain weight, a 3-set and a 4-set of a related weight in a coherent way.
- Procedures P_j for $2 \leq j \leq 4$ which are called in in reverse order in the following manner.

- Our overall goal begins with, say $P_4(\frac{7}{8}, \frac{1}{8})$
- Load $\frac{7}{8}$ beyond $\gamma(m_4, s_0)$ initially in chunks of $\frac{1}{8}$, this being a 4-set.

- Our overall goal begins with, say $P_4(\frac{7}{8}, \frac{1}{8})$
- Load $\frac{7}{8}$ beyond $\gamma(m_4, s_0)$ initially in chunks of $\frac{1}{8}$, this being a 4-set.
- The procedure P_j ($2 \leq j \leq 4$) enumerates a j-set C_j. The construction begins by calling P_4, which calls P_3 several times, and so on down to P_2, which enumerates the 2-set C_2 and a KC set L of axioms $\langle q, n \rangle$.

- Our overall goal begins with, say $P_4(\frac{7}{8}, \frac{1}{8})$
- Load $\frac{7}{8}$ beyond $\gamma(m_4, s_0)$ initially in chunks of $\frac{1}{8}$, this being a 4-set.
- The procedure P_j ($2 \leq j \leq 4$) enumerates a j-set C_j. The construction begins by calling P_4, which calls P_3 several times, and so on down to P_2, which enumerates the 2-set C_2 and a KC set L of axioms $\langle q, n \rangle$.
- Each procedure P_j has rational parameters $q, \beta \in [0, 1]$. The goal q is the weight it wants C_j to reach, and the garbage quota β is how much it is allowed to waste.

- In the impossible construction, where there was only one m, the goal was $\frac{7}{8}$ and the β evolved with time. The same thing happens here. P_4's goal never changes, and hence can never be met lest U use too much quanta. Thus A cannot compute B.
- The main idea is that procedures P_j will ask that procedures P_i for $i < j$ do the work for them, with eventually P_2 "really" doing the work, but the the goals of the P_i are determined inductively by the garbage quotas of the P_j above.

- In the impossible construction, where there was only one m, the goal was $\frac{7}{8}$ and the β evolved with time. The same thing happens here. P_4's goal never changes, and hence can never be met lest U use too much quanta. Thus A cannot compute B.
- The main idea is that procedures P_j will ask that procedures P_i for $i < j$ do the work for them, with eventually P_2 "really" doing the work, but the the goals of the P_i are determined inductively by the garbage quotas of the P_j above.
- Then if the procedures are canceled before completing their tasks then the amount of quanta wasted is acceptably small.

- $P_4(\frac{7}{8}, \frac{1}{8})$. Its action:
  1. Choose m_4 large.
  2. Wait until $\Gamma^A(m_4) \downarrow$.

- $P_4(\frac{7}{8}, \frac{1}{8})$. Its action:
  1. Choose m_4 large.
  2. Wait until $\Gamma^A(m_4)\downarrow$.
- Then P_4 will call $P_3(2^{-4}, 2^{-5})$.

- $P_4(\frac{7}{8}, \frac{1}{8})$. Its action:
  1. Choose m_4 large.
  2. Wait until $\Gamma^A(m_4)\downarrow$.
- Then P_4 will call $P_3(2^{-4}, 2^{-5})$.
- Note that here the idea is that P_4 is asking P_3 to enumerate the 2^{-4}'s which are the current quanta bits that P_4 would like to load beyond m_4's current Γ-use.

- $P_4(\frac{7}{8}, \frac{1}{8})$. Its action:
  1. Choose m_4 large.
  2. Wait until $\Gamma^A(m_4) \downarrow$.
- Then P_4 will call $P_3(2^{-4}, 2^{-5})$.
- Note that here the idea is that P_4 is asking P_3 to enumerate the 2^{-4}'s which are the current quanta bits that P_4 would like to load beyond m_4's current Γ-use.
- If the Γ-use of m_4 changes, then we will go back to the beginning.

- $P_3(2^{-4}, 2^{-5})$ will:
  1. Choose m_3 large.
  2. Wait until $\Gamma^A(m_3)\downarrow$.

- $P_3(2^{-4}, 2^{-5})$ will:
  1. Choose m_3 large.
  2. Wait until $\Gamma^A(m_3)\downarrow$.
- Then it will invoke $P_2(2^{-5}, 2^{-6})$, etc.

- $P_3(2^{-4}, 2^{-5})$ will:
    1. Choose m_3 large.
    2. Wait until $\Gamma^A(m_3)\downarrow$.
- Then it will invoke $P_2(2^{-5}, 2^{-6})$, etc.
- It is only $P_2(2^{-5}, 2^{-6})$,

- $P_3(2^{-4}, 2^{-5})$ will:
  1. Choose m_3 large.
  2. Wait until $\Gamma^A(m_3)\downarrow$.
- Then it will invoke $P_2(2^{-5}, 2^{-6})$, etc.
- It is only $P_2(2^{-5}, 2^{-6})$,
- That is, load 2^{-5} beyond $\gamma(m_2, s)$ in lots of 2^{-6}.

- $P_3(2^{-4}, 2^{-5})$ will:
  1. Choose m_3 large.
  2. Wait until $\Gamma^A(m_3)\downarrow$.
- Then it will invoke $P_2(2^{-5}, 2^{-6})$, etc.
- It is only $P_2(2^{-5}, 2^{-6})$,
- That is, load 2^{-5} beyond $\gamma(m_2, s)$ in lots of 2^{-6}.
- Only P_2 puts numbers into B to induce an A-change.

Summary

In general, the inductive procedures work the same way. Whilst waiting, if uses change, then we will initialize the lower procedures, reset their garbages to be ever smaller, but not throw away any work that has been successfully completed. Then in the end we can argue by induction that all tasks are completed.

Similar methods allow for us to show the following

THEOREM (NIES)

All K-trivials are superlow $A' \equiv_{tt} \emptyset'$, and are tt-bounded by c.e. K-trivials.

Thus triviality is essentially an "enumerable" phenomenom.

- This and the results to follow use a similar trick:
- Suppose that we play the decanter game on a K-trivial. Then the procedures won't return.

- This and the results to follow use a similar trick:
- Suppose that we play the decanter game on a K-trivial. Then the procedures won't return.
- Now suppose that we want to show a K-trivial A is (super-)low.

- This and the results to follow use a similar trick:
- Suppose that we play the decanter game on a K-trivial. Then the procedures won't return.
- Now suppose that we want to show a K-trivial A is (super-)low.
- We want to decide should we believe some computation $\Phi_e^A(e) \downarrow [s]$. Is $A_s \upharpoonright \varphi_e(x,s) = A\varphi_e(e)$?

- The idea is that we want to load enough quanta beyond $\varphi_e(x, s)$ that the computation is believable, or it costs the opponent a lot to show us wrong. Thus we believe, i.e. change from believing $\downarrow$ to $\uparrow$, when the inductive procedures below load enough quanta.
- This should be thought of as an ω branching tree of possibilities, one for each argument e.
- Each of the possibilities is given some quota varying with s, but is "like" $2^{-(e+1)}$, also with an inductive trash quota.
- At the outcome assuming that the procedure does not return, we begin another attampt to compute the jump of A working on computations $\Phi_j^A(j)$, for $j > \varphi(e, s)$. These are given each a quota, the total being essentially the trash of the procedure above.
- Nies calls the procedure which does not return the golden run, and it is this that constructs the correct procedure computing the jump.

There are other antirandomness notions.

Definition (Kučera and Terwijn)

We say A is low for randomness iff the reals Martin-Löf random relative to A are exactly the Martin-Löf random reals.

Definition (Hirschfeldt, Nies, Stephan)

A is a a base of a cone of randomness iff $A \leq_T B$ with B A-random.

THEOREM

The following are equivalent.

(I) *(Nies) A is low for randomness.*

(II) *(Hirschfeldt and Nies) A is K-low in that* $K^A =^+ K$.

(III) *(Hirschfeldt, Nies, Stephan) A is a base of a cone of randomness.*

(IV) *(Downey, Nies, Weber, Yu+Nies, Miller) A is low for weak-2-randomness.*

QUESTIONS AND A PROPER SUBCLASS

It is open if this is the same as a number of other "cost function" classes such as the reals which are Martin-Löf cuppable to $\emptyset'$. (Nies)

It is known there is a proper subclass defined by cost function.

DEFINITION (NIES)

Let h be an order. We say that A is jump traceable for the order h iff there is a computable collection of c.e. sets $W_{g(e)}$ with $|W_{g(e)}| < h(e)$ and $J^A(e) \in W_{g(e)}$. A is strongly jump traceable iff it is jump traceable for every computable order.

Theorem (Nies)

A is K-triv implies that there is an order h (roughly $n \log n$) relative to which A is jump traceable.

Theorem (Fugiera, Nies, Stephan)

Noncomputable sjt c.e. sets exist.

Theorem (Nies)

A is K-triv implies that there is an order h (roughly $n \log n$) relative to which A is jump traceable.

Theorem (Fugiera, Nies, Stephan)

Noncomputable sjt c.e. sets exist.

Theorem (Cholak, Downey, Greenberg)

The c.e. sjt's are a proper subclass of the K-trivials.

- Roughly need orders $\log \log n$. Is there a combinatorial characterization?

Lots of ignored work

There is a lot of material on lowness for e.g. Schnorr, Kurtz and computable randomness. For example:

Theorem (Nies)

No noncomputable set is low for computable randomness

Theorem (Terwijn-Zambella, Nies, etc)

A is computably traceable iff A is low for Schnorr random.

Computably traceable is roughly uniformly hyperimmune free.

Theorem (Downey-Griffiths, +Stephan-Yu)

The low for Kurtz random properly contain the low for Schnorr randoms, and are properly contained in the hyperimmune free degrees.