A left-c.e. real not computable by any Omega number with low redundancy

Nan Fang

Universität Heidelberg
Germany

CCR 2017, Mysore, India

computational redundancy

When restricting the use function of a Turing reduction, we could get reducibilities of different strength, like

- *wtt*-reducibility: Turing reducibility where the use function is bounded by a computable function.
- *cl*-reducibility: Turing reducibility where the use function is bounded by $n + c$ for some constant c.

etc. Yet there is another view to look at those reducibilities:

If a set A is Turing computable from a set B with use function bounded by $n + h(n)$, or equivalently, $A \upharpoonright n$ is computable from $B \upharpoonright n + h(n)$, we say A is computable from B with *redundancy* h, or h is the computational redundancy from B to A.

Here redundancy means that during the computation, some extra bits of information of the oracle is required, compared to the bits of information produced.

computational redundancy

For some reducibility notions, considering their redundancies we have the following table:

reducibility	use bound	redundancy
wtt	computable	computable
lbT	linear	linear
cl	n+constant	constant
ibT	n	0

As most of the time we care only computable and increasing use bounds, we will assume all redundancy functions are computable and nondecreasing.

Let $h : \mathbb{N} \mapsto \mathbb{N}$ be a computable and nondecreasing function.

- If $\sum_i 2^{-h(i)} = \infty$, we say h is a *low* redundancy.
- If $\sum_i 2^{-h(i)} < \infty$, we say h is a *high* redundancy.

Kučera-Gács theorem and redundancy

Theorem (Kučera; Gács)

Every set is wtt-reducible to a Martin-Löf random set.

A careful analysis gives us the following smallest redundancy approximations required in their proofs:

- Kučera's redundancy: $n \log n$
- Gács's redundancy: $\sqrt{n} \log n$

Kučera-Gács theorem and redundancy

However, both redundancies turned out to be not optimal. And there exist optimal redundancy.

Theorem (Barmpalias, Lewis-Pye, Teutsch 2017)

Every set is computable from some Martin-Löf random set with any given high redundancy.

Theorem (Barmpalias, Lewis-Pye 2016)

There is a set not computable from any Martin-Löf random set with any given low redundancy.

compute left-c.e. reals from Ω number

- When focusing only on left-c.e. reals, it is easy to prove a local version of the Kučera-Gács theorem.
- Though it doesn't follow directly from the global version, the following two theorem indicated the same optimal redundancy required.

Theorem (Barmpalias, F., Lewis-Pye 2016)

*Every left-c.e. real is computable from **any** left-c.e. Martin-Löf random real with any given high redundancy.*

Theorem (Barmpalias, F., Lewis-Pye 2016)

For any left-c.e. Martin-Löf random real α, there is a left-c.e. real which cannot be computed by α with any low redundancy.

compute left-c.e. reals from Ω number

However, the second theorem for the local case is slightly different from the global case, and there is the following theorem for *cl*-reducibility.

Theorem (Barmpalias, Lewis-Pye 2006)

There is a left-c.e. real not cl-computable from any left-c.e. Martin-Löf random real.

It is natural to ask whether we can get the same result as above for all computations with low redundancies.

Fortunately, the answers is yes!

Theorem (F. 2017)

There is a left-c.e. real not computable from any left-c.e. Martin-Löf random real with any given low redundancy.

proof of the theorem

We need to construct a left-c.e. real α satisfies following requirements:

$$\mathcal{R}_i : \alpha = \Phi_i^{\beta_i} \Rightarrow \beta_i \text{ is not Martin-Löf random}$$

where $\langle \Phi_i, \beta_i \rangle$ lists all pairs of a Turing functional with some low redundancy and a left-c.e. real in the unit interval.

$\mathcal{R}_i$ can be broken into

$$\mathcal{Q}_{\langle i,j \rangle} : \alpha = \Phi_i^{\beta_i} \Rightarrow \beta_i \in [E_j^i]$$

where $[E_j^i]$ is the jth member of some Martin-Löf test we need to construct for $\mathcal{R}_i$.

It suffices if we make $E_j^i \in 2^{<\omega}$ computable and $\forall i \ \mu(E_j^i) \leq 2^{-j}$.

game $G(h, m, \epsilon)$

For a redundancy h, an integer $m > 0$ and a real $\epsilon > 0$, the game $G(h, m, \epsilon)$ between two players A and B is as following:

1. Player A controls a real α on positions $[m, +\infty)$ and a set E. Player B controls β. The game starts with all bits of α on positions $[m, +\infty)$ being 0, E being empty and $\beta \geq 0$.
2. At every run, player A can change some bits of α on positions $[m, +\infty)$ or put some string into E.
3. Whenever α is changed at position t(we say A issued a coding instruction at position t), B should change β at some position $\leq t + h(t)$.
4. Whenever some initial segment of the current β of length t is enumerated into E(we say A issued a test instruction at position t), B should change β at some position $\leq t$.
5. The measure of the set E should not exceed ϵ.
6. At every run, the values of α and β should not be decreased.

game $G(h, m, \epsilon)$

Player A wins the game $G(h, m, \epsilon)$ if after finitely many movements $\beta \geq 1$.

We can prove that player A has a winning strategy.

Lemma

For any low redundancy function h, any integer $m > 0$ and any real $\epsilon > 0$, player A in the game $G(h, m, \epsilon)$ always has a winning strategy. Moreover, the strategy only changes α on some positions interval $[m, m']$, and m' is computable from (h, m, ϵ).

We will call $[m, m']$ the sufficient interval of the strategy for the corresponding game.

proof of the theorem given the lemma

To each $Q_{\langle i,j\rangle}$ we assign an interval $I_{\langle i,j\rangle}$ in a priority order:

- Suppose $I_f(f < \langle i,j\rangle)$ are defined and m is the largest number in $I_{\langle i,j\rangle - 1}$.
- Consider the game $G(h_i, m+1, 2^{-j})$. By the lemma there is a winning strategy for A. We define $I_{\langle i,j\rangle}$ to be the sufficient interval, which is computable.
- We call the corresponding strategy $P_{\langle i,j\rangle}$, and let E^i_j the set E produced by the strategy.

At stage s, $Q_{\langle i,j\rangle}$ requires attention if:

- $\Phi_i^{\beta_i}[s](n) = \alpha[s](n)$ for all $n \leq \max I_{\langle i,j\rangle}$
- and $\beta_i[s] \notin [E^i_j[s]]$.

proof of the theorem given the lemma

Construction: Let $\alpha_0 = 0$. At stage s, find the least number $e \leq s$ where Q_e requires attention.

- If exists, for $t > e$ let all bits of α on I_t be 0 and initialize all the strategy P_t. Then perform the next step of the strategy P_e.
- Otherwise, move to stage $s + 1$ directly.

Verification: Suppose requirement $Q_{\langle i,j \rangle}$ is the least one not satisfied.

- Then the reduction $\alpha = \Phi_i^{\beta_i}$ is total and $\beta_i \in [0, 1)$ is random.
- B should have followed all the coding instructions and test instructions issued by $P_{\langle i,j \rangle}$.
- As $P_{\langle i,j \rangle}$ is a winning strategy for A in the game $G(h_i, m, 2^{-j})$, then $\beta_i \geq 1$, which is a contradiction.

Thus, all the requirements $Q_{\langle i,j \rangle}$ will then be satisfied. And we can check that α is a left-c.e. real.

proof of the lemma

For $t \geq m$, we design a procedure $P(n,t)$ as part of a strategy for player A in the game $G(h,m,\epsilon)$ which starts with α_0, β_0 and end with α_f, β_f where

1. $\alpha_0(i) = 0$ for $i \geq t$.
2. $\alpha_f(i) = \alpha_0(i)$ for $i < t$.
3. $\beta_f \geq \beta_0 \upharpoonright g(t) + 2^{n-t}$.

For example, a configuration for $P(3,t)$ could be:

		1		$t-3$			t			
$\alpha_0 =$	0.	0	$\cdots$	0	0	0	0	0	0	$\cdots$
$\beta_0 =$	0.	0	$\cdots$	0	0	0	0	0	0	$\cdots$
$\alpha_f =$	0.	0	$\cdots$	0	0	0	1	1	1	$\cdots$
$\beta_f =$	0.	0	$\cdots$	1	0	0	0	0	0	$\cdots$

proof of the lemma

The cost $C(n,t)$ of the procedure $P(n,t)$ is defined to be the measure of the strings enumerated into the set E during the procedure.

It is clear that $P(n,n)$ will be a winning strategy for A if $C(n,n) < \epsilon$ for some $n \geq m$.

We will define $P(n,t)$ inductively. At first, we fix some integer $m_0 \gg m,n$.

proof of the lemma

Let $g(n) = n + h(n)$. As the induction basis, we define $P(n, t)$ for $t = m_0 + 1, ..., g(m_0) + n$ as follows:

(1) Change α at position t from 0 to 1.
(2) Issue $2^{n+h(t)} - 1$ many times test instructions at position $g(t)$.

It is easy to check that these $P(n, t)$s are well defined.

And $C(n, t) = 2^{n-t} - 2^{-g(t)}$.

proof of the lemma

One example is as following:

		1		$t-n$			t				$g(t)$		
$\alpha_0 =$	0.	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	0	0	$\cdots$
$\beta_0 =$	0.	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	0	0	$\cdots$
$\alpha_1 =$	0.	0	$\cdots$	0	0	$\cdots$	1	0	$\cdots$	0	0	0	$\cdots$
$\beta_1 =$	0.	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	1	0	$\cdots$
						$\vdots$							
$\alpha_f =$	0.	0	$\cdots$	0	0	$\cdots$	1	0	$\cdots$	0	0	0	$\cdots$
$\beta_f =$	0.	0	$\cdots$	1	0	$\cdots$	0	0	$\cdots$	0	0	0	$\cdots$

proof of the lemma

Suppose we already have procedures $P(n,i)$ for $i = t+1, ..., g(t)+n-1$, we define procedure $P(n,t)$ for $t \leq m_0$ as follows:

(p_1)	Run $P(n, t+1)$.
(a_1)	Change α at position t from 0 to 1.
(p_2)	Run $P(n, t+2)$.
(a_2)	Change α at position $t+1$ from 0 to 1.
...	...
$(p_{h(t)+n-1})$	Run $P(n, g(t)+n-1)$.
$(a_{h(t)+n-1})$	Change α at position $g(t)+n-2$ from 0 to 1.
$(a_{h(t)+n})$	Change α at position $g(t)+n-1$ from 0 to 1.
(f)	Issue $(2^{-g(t)} - \sum_{i=t+1}^{g(t)+n-1} 2^{-i-h(i)}) \cdot 2^{g(g(t)+n-1)}$ many times test instructions at position $g(g(t)+n-1)$.

Noted that when α was changed at position k from 0 to 1 at all positions $l > k$ of α can be initialized to 0.

proof of the lemma

		t-n	t-n+1		t	t+1		g(t)-1	g(t)		g(t)+n-2	g(t)+n-1	
$\alpha_0 =$	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$
$\beta_0 =$	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$
$\alpha_{p_1} =$	$\cdots$	0	0	$\cdots$	0	1	$\cdots$	?	?	$\cdots$	?	?	$\cdots$
$\beta_{p_1} =$	$\cdots$	0	1	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$
$\alpha_{a_1} =$	$\cdots$	0	0	$\cdots$	1	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$
$\beta_{a_1} =$	$\cdots$	0	1	$\cdots$	0	0	$\cdots$	0	1	$\cdots$	0	0	$\cdots$
						$\vdots$							
$\alpha_{a_{h(t)+n-1}} =$	$\cdots$	0	0	$\cdots$	1	1	$\cdots$	1	1	$\cdots$	1	0	$\cdots$
$\beta_{a_{h(t)+n-1}} =$	$\cdots$	0	1	$\cdots$	1	1	$\cdots$	1	1	$\cdots$	?	?	$\cdots$
$\alpha_{a_{h(t)+n}} =$	$\cdots$	0	0	$\cdots$	1	1	$\cdots$	1	1	$\cdots$	1	1	$\cdots$
$\beta_{a_{h(t)+n}} =$	$\cdots$	0	1	$\cdots$	1	1	$\cdots$	1	1	$\cdots$	?	?	$\cdots$
$\alpha_f =$	$\cdots$	0	0	$\cdots$	1	1	$\cdots$	1	1	$\cdots$	1	1	$\cdots$
$\beta_f =$	$\cdots$	1	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$

proof of the lemma

We can also check that the induction steps of $P(n,t)$ are well defined, and

$$C(n,t) = \sum_{i=t+1}^{g(t)+n-1} C(n,i) + 2^{-g(t)} - \sum_{i=t+1}^{g(t)+n-1} 2^{-g(i)}$$

Let $D(n,i) = C(n,i) - 2^{-g(i)}$, we can get $\forall t \in [m, m_0)$

$$D(n,t) \leq \prod_{i=t}^{m_0-1} (1 - 2^{-n-h(i)})$$

proof of the lemma

- Find the least $n \geq m$ such that $2^{-n-h(n)} < \epsilon/2$.
- As $\sum_{i \in \mathbb{N}} 2^{-h(i)} = \infty$, we also have $\sum_{i \geq n} 2^{-n-h(i)} = \infty$. By a theorem from analysis we have $\prod_{i \geq n}(1 - 2^{-n-h(i)}) = 0$.
- Find $n_0 > n$ such that $\prod_{i=n}^{n_0-1}(1 - 2^{-n-h(i)}) < \epsilon/2$.

We define $P(n, i)$ for $i = n_0 + 1, ..., n_0 + h(n_0) + n$ as the induction basis, and $P(n, i)$ for $i \in [n, n_0]$ as in the induction process. Then $D(n, n) \leq \prod_{i=n}^{n_0-1}(1 - 2^{-n-h(i)}) < \epsilon/2$. Thus, $C(n, n) = D(n, n) + 2^{-n-h(n)} < \epsilon$. Therefore $P(n, n)$ is a winning strategy for A in the game $G(h, m, \epsilon)$.