THE SETTLING TIME REDUCIBILITY ORDERING AND Δ^0_2 SETS

BARBARA F. CSIMA

Abstract. The Settling Time reducibility ordering gives an ordering on computably enumerable sets based on their enumerations. The $<_{st}$ ordering is in fact an ordering on c.e. *sets*, since it is independent of the particular enumeration chosen. In this paper we show that it is not possible to extend this ordering in an approximation-independent way to Δ^0_2 sets in general, or even to n-c.e. sets for any fixed $n \geq 3$.

1. Introduction

The settling time reducibility ordering, $<_{\mathrm{st}}$, on c.e. sets was first introduced by Nabutovsky, Weinberger and Soare (see [5]), for application to differential geometry. The ordering extends the idea of domination to a partial ordering on c.e. sets. For an overview of what is known about $<_{\mathrm{st}}$, see [2].

For Computability Theory, we follow the notation of Soare's *Recursively Enumerable Sets and Degrees* [4] and new notation from Soare's *Computability Theory and Applications* [6], which we also define. See also Cooper's *Computability Theory* [1] for a modern treatment of the subject.

We first recall the definition of a dominant function. We write "$\forall^\infty x$" for "for all but finitely many x".

Definition 1.1. A function g is said to be *dominant* if for every computable function f, $(\forall^\infty x)[g(x) > f(x)]$.

We now give the definition of the settling function associated to an enumeration of a c.e. set.

Definition 1.2. For every computably enumerable (c.e.) set A and associated enumeration $\{A_s\}_{s\in\omega}$,we define the *settling (or modulus) function*: $m_A(x) = (\mu s)[A_s \restriction x = A \restriction x]$ where $A \restriction x = \{y \leq x \mid y \in A\}$.

The settling time reducibility ordering on c.e. sets is defined as follows.

Definition 1.3. For c.e. sets A and B, with enumerations $\{A_s\}_{s\in\omega}$ and $\{B_s\}_{s\in\omega}$, we say A *settling time dominates* B and write $A >_{st} B$ iff

$$(\forall \text{ computable } f)(\forall^\infty x)[\ m_A(x)\ >\ f(m_B(x))\ ].$$

Andre Nies (unpublished, see [3]) showed that this is independent of the particular enumerations chosen. Thus $<_{\mathrm{st}}$ is indeed an ordering on c.e. *sets*.

B. Csima was partially supported by Canadian NSERC Discovery Grant 312501.

Since Δ^0_2 sets also have nice approximations, a question I often hear after discussing the $<_{\rm st}$ ordering is "What about Δ^0_2 sets?" In the past, I have always brushed off such questions with the response "Oh, for Δ^0_2 sets it doesn't make sense because you can just jiggle on a spot to push up the settling time." However, after my talk at the *Special Session on Computability and Mathematical Structure, CiE 2007*, the questioning was more persistent: "What about n-c.e. sets for fixed n?" "What if you consider the computation function instead of the settling function?" In this paper, I address these questions.

For c.e. sets, the stage by which an approximation has settled up to x is the same as the first stage that it was correct up to x. Δ^0_2-approximations do not have this property.

We extend the definition of settling function to Δ^0_2 sets as follows.

Definition 1.4. For any Δ^0_2 set A with approximation $\{A_s\}_{s\in\omega}$, define the *settling function* for A, $m_A(x)$, to be the least stage after which the approximation has settled up to x. That is,

$$m_A(x) = (\mu s)(\forall t \geq s)[A_t{\restriction} x = A{\restriction} x].$$

We will see in Section 2 that this definition does not lead to an approximation-invariant ordering for Δ^0_2 sets. This is true in a strong sense, as we will build for any $n \geq 2$ a properly n-c.e. set with two different n-c.e. enumerations, one settling time dominating the other.

However, it might make more sense to consider for Δ^0_2 sets the *computation function*, that gives the first stage after x that an enumeration is correct up to x, and see if an approximation-independent ordering can be introduced in this way.

We recall the definition of the computation function.

Definition 1.5. For any Δ^0_2 set A with approximation $\{A_s\}_{s\in\omega}$, define the *computation function* for A, $C_A(x)$, to be the least stage after x where the approximation is correct up to x. That is,

$$C_A(x) = (\mu s \geq x)[A_s{\restriction} x = A{\restriction} x].$$

The computation function is perhaps more natural to consider than the settling function, since for Δ^0_2 sets A, $C_A \equiv_T A$, while this is not necessarily true for m_A.

So we might try to define A *computation time dominates* B ($A >_{ct} B$) if for all computable functions f, C_A dominates $f \circ C_B$.

In Section 3 we will see that for any $n \geq 3$ there exists a properly n-c.e. set A with two different n-c.e. approximations, one of which computation time dominates the other. So computation time domination will not be independent of the approximation chosen.

Indeed, we will construct a c.e. set A that has a 3-c.e. approximation the computation function of which its c.e. approximation will computation time dominate.

Hence, the question that remains open will be: Is the computation time domination ordering approximation-independent for 2-c.e. sets, where we allow only 2-c.e. approximations?

Convention 1.6. We let $\{\varphi_n\}_{n\in\omega}$ be a list of all non-decreasing partial computable functions with domain an initial segment of ω. We assume that if $\varphi_{n,s}(x)\downarrow$, then $n, x < s$, and $\varphi_{n,s}(y)\downarrow$ for all $y < x$.

We note that "for all computable f, for almost every x, $m_A(x) > f \circ m_A(x)$", holds exactly if it holds with "for all total φ_n" replacing "for all computable f". The same is true for C_A in place of m_A.

2. The settling function

We now show that the settling function m_A does not lend itself to defining an approximation-invariant domination reducibility on Δ^0_2 sets.

Theorem 2.1. *For any $n \geq 2$ there exists a properly n-c.e. set A with two n-c.e. approximations $\{A_s\}$ and $\{\tilde{A}_s\}$ such that for any computable function f, $(\forall^\infty x)[m_{\tilde{A}}(x) \geq f \circ m_A(x)]$. Indeed, such an A can be found in every proper n-c.e. degree.*

Proof. This is very similar to the construction to build two c.e. sets $\tilde{A}$ and A such that $\tilde{A} >_{\text{st}} A$. The only difference now is that since we are working with Δ^0_2-approximations, we can push up the values of one approximation, $m_{\tilde{A}}(x)$, by just inserting and then removing a number less than x into $\tilde{A}$, and so still approximating the same set as A.

Let D be any properly n-c.e. set, together with an n-c.e. approximation $\{D_s\}_{s\in\omega}$.

Let $d_0 = 0$. For $x \geq 0$, let $d_{x+1} = d_x + (n+1)(x+1)$. We will use d_x to code whether $x \in D$. That is, we will have $d_x \in A \iff x \in D$, and we will $y \notin A$ if $y \neq d_x$ for any x. Hence A and D will be m-equivalent.

We now construct the approximations A and $\tilde{A}$ by stages. We will use helper numbers c_x to keep track of our current guess at the settling time of d_x in A, and l_x to keep track of how many times c_x has changed.

Stage 0: Let $l_x = 0$ for all x.

Stage $s+1$: Step 1. If $x \in D_{s+1} - D_s$, enumerate d_x into A_{s+1} and $\tilde{A}_{s+1}$. Set $c_x = s+1$, and increment l_x by 1. Similarly, if $x \in D_s - D_{s+1}$, remove d_x from A_{s+1} and $\tilde{A}_{s+1}$. Set $c_x = s+1$, and increment l_x by 1.

Step 2. If $\varphi_i(c_{x+1})$ converged at stage $s+1$ for some $i \leq x$, enumerate $d_x + l_x n + (i+1)$ into $\tilde{A}_{s+1}$, and then remove $d_x + l_x n + (i+1)$ from $\tilde{A}_{s+2}$ (note that this is the only circumstance under which $d_x + l_x n + (i+1)$ will be enumerated/removed from $\tilde{A}$).

This completes the construction.

Notice that if $y \neq d_x$ for any x, then y is never enumerated into A, and may only enter and then exit $\tilde{A}$. It follows that since we were following the n-c.e. approximation for D to decide when d_x is in A and $\tilde{A}$, and $n \geq 2$, that A and $\tilde{A}$ are n-c.e. Since $x \in D \iff d_x \in A$, and since D is properly n-c.e. , it follows that the approximations A and $\tilde{A}$ must also be properly n-c.e. , as the sequence d_x is computable, and so any m-c.e. approximation to A would yield an m-c.e. approximation to D.

Let f be a total computable function. We must show that $(\forall^\infty x)[m_{\tilde{A}}(x) > f(m_A(x))]$. It suffices to show that for a.e. x, if x entered or exited A at a stage s, some $z \leq x$ entered or exited $\tilde{A}$ at a stage greater than $f(s)$. Let i be such that $f = \varphi_i$. Let $x > d_{i+1}$, and suppose x entered or exited A at stage s. Then by construction, $x = d_{y+1}$ for some $y > i$, and at stage s, we would have set $c_{y+1} = s$. At the stage t when $\varphi_i(c_{y+1})$ converged, $d_y + l_y n + (i+1)$ was enumerated and then removed from A. Note that $f(s) = \varphi_i(s) < t$, and that $d_y + l_y n + (i+1) < d_{y+1}$. □

3. The computation function

Now we address the computation function, C_A.

Theorem 3.1. *There exists a c.e. set A with a 3-c.e. approximation $\{A_s\}_{s\in\omega}$, whose computation function we will denote C_A, and a c.e. approximation $\{\tilde{A}\}_{s\in\omega}$, whose computation function we will denote $C_{\tilde{A}}$, such that for every computable function f, $(\forall^\infty x)[C_{\tilde{A}}(x) > f \circ C_A(x)]$.*

Proof. The basic idea is that with a 3-c.e. approximation, we can *tempt* by enumerating and then immediately removing a number from one approximation, thus setting a possible computation value that we may return to, and then later still have the opportunity to enumerate that number into both approximations, returning to an old computation in the first case, and creating a new one in the second. For example, enumerate 0 into A at stage s, remove it at stage $s+1$, and do nothing to $\tilde{A}$. Then the approximations to A and $\tilde{A}$ will agree at stage $s+1$, and we still have the option to enumerate 0 into either approximation at a later stage and keep A and $\tilde{A}$ 3-c.e. . Now if $\varphi_0(s)\downarrow$ at some later stage t, then enumerate 0 into A and $\tilde{A}$ at stage t. Thus we will have $C_{\tilde{A}}(0) = t > \varphi_0(s) = \varphi_0(C_A(0))$.

For the general case we will deal with larger and larger blocks, that aim to please more and more φ_e. The blocks will be separated into sub-blocks, because as we are waiting to find out whether φ_0 converges on our tempting stage for 0, we will want to tempt also for the sake of φ_1. So we tempt by enumerating and removing some number, x, from A, but that temptation is only useful to us while 0 remains out of A. So if we later enumerate 0 into A and $\tilde{A}$, then the temptation we made for the sake of φ_1 will have been useless. In this case we will pass down to a smaller number $y < x$, and tempt again by enumerating and removing y from A. Since we will be building $\tilde{A}$ to be c.e. , we know that on an initial segment of size n, we can pass through at most $n+1$ configurations.

We separate ω into consecutive finite blocks B_0, B_1,..., where $|B_n| = (n+1)(1+\Sigma_{k<n}|B_k|)$. So $B_0 = \{0\}$, $B_1 = \{1,2,3,4\}$, ...

We consider B_n as consisting of $(1+\Sigma_{k<n}|B_k|)$ many sub-blocks, $B_{n,0}$, ..., $B_{n,\Sigma_{k<n}|B_k|}$, each of size $n+1$.

If at stage s in the construction we say to *tempt with block $B_{n,l}$*, then we mean to do the following. We have $B_{n,l} = \{b_0 < ... < b_n\}$. Enumerate b_n into A at stage s, enumerate b_{n-1} into A at stage $s+1$, and so on, enumerating b_{n-k} into A at stage $s+k$ for each $0 \leq k \leq n$. Then at stage $s+n+1$, remove all of $b_0, ..., b_n$ from A.

In the construction, at any given stage, in each block B_n there will be at most one active sub-block. A sub-block will be activated at a stage when it is used for tempting. The first sub-block to be activated will be the greatest one, $B_{n,\Sigma_{k<n}|B_k|}$. Future sub-blocks $B_{n,l}$ will only be activated if the sub-block $B_{n,l+1}$ was the previously active sub-block for B_n. When the sub-block $B_{n,l}$ is activated, the sub-block $B_{n,l+1}$ will be deactivated. A number x will only be enumerated or removed from A or $\tilde{A}$ if it belongs to an active sub-block.

During the construction, we will use numbers c_n to keep track of the stage when the last temptation into block B_n was completed. That is, if at stage s we activate a sub-block $B_{n,l}$ and tempt with it, then the temptation would be completed at stage $s+n+1$, so we would set $c_n = s+n+1$.

Note that in each step of the construction, we will pass through many stages, as we tempt with various blocks.

Construction:

Step 0 : Let $stage(0) = 0$, and $stage(1) = 1$.

Step t: Set $s = stage(t)$. If $\varphi_{e,s}(c_n) \downarrow$ for some $e \leq n$ for which B_n has an active sub-block (and if $\varphi_{e,stage(t-1)}(c_n) \downarrow$), then if $B_{n,l}$ is that active sub-block, choose the largest $x \in B_{n,l}$ such that $x \notin A$, and enumerate x into both A and $\tilde{A}$. Reset $s := s + 1$. For each $m > n$, in turn, if B_m had an active sub-block, $B_{m,k}$, then tempt in A with $B_{m,k-1}$, and declare $B_{m,k-1}$ to be the active sub-block of B_m. Redefine $c_m = s + m + 1$, that is, c_m should be the stage where we finished tempting. Reset $s := s + m + 1$.

For the least n such that B_n does not have an active sub-block, if $s > \max\{B_{n+1}\}$, tempt in A with $B_{n,\Sigma_{k<n}|B_k|}$, declare $B_{n,\Sigma_{k<n}|B_k|}$ to be active, and define $c_n = s + n + 1$. Reset $s := s + n + 1$.

Let $stage(t + 1) = s$.

This completes the construction.

We now note some facts about the construction. First, $\tilde{A}$ is c.e. , since we only ever enumerate into it, and never remove anything from it. Second, $A = \tilde{A}$. This is because we only enumerate into $\tilde{A}$ if we simultaneously enumerate into A, and because if we enumerate into A without enumerating into $\tilde{A}$, then this is during a temptation, and whatever is enumerated is removed from A at the end of the temptation.

Notice that we wait until after stage $\max\{B_{n+1}\}$ before we do anything with any numbers in block B_n. This is so that any action we take with the numbers in block B_n will have an effect on the computation values for the numbers in block B_{n+1}.

Note that c_m is only redefined, and the active sub-block of B_m shifted, if some number $x \in B_n$ enters $\tilde{A}$ for some $n < m$. Since A is c.e. , this can happen at most $\Sigma_{k<n}|B_k|$ many times, and so there is always a lower sub-block to shift to if needed, and c_m is only redefined finitely often. Also note that whenever we re-define c_m, we also shift the active sub-block to a brand new one, none of whose members are in $\tilde{A}$. So since each sub-block has size $m + 1$, if we notice that $\varphi_{e,s}(c_m) \downarrow$ for some $e \leq m$, there must be an available x in the currently active sub-block to enumerate into $\tilde{A}$.

Lemma 3.2. *For all $x \in B_n$, if φ_e is total and $e < n$ then $C_{\tilde{A}}(x) > \varphi_e(C_A(x))$.*

Proof. Within each block B_n, the numbers are enumerated into $\tilde{A}$ in decreasing order. Let y be the least number enumerated into $\tilde{A}$ from B_n. Since we enumerated in decreasing order, it is also the last. Note that since φ_e is total, such y must exist. Then for $x \geq y$, $C_{\tilde{A}}(x) = C_{\tilde{A}}(y)$. Let k be such that $y \in B_{n,k}$. Since φ_e is total, $\varphi_e(c_n)$ must have converged at some point. If this happened after we had defined c_n, then we would have enumerated some $z \in B_{n,k}$ into $\tilde{A}$ at a stage $t > \varphi_e(c_n)$, and since $y \leq z$ was enumerated later or at the same moment, $C_{\tilde{A}}(y) \geq t > \varphi_e(c_n)$. If $\varphi_e(c_n)$ had already converged at some earlier stage, then certainly $C_{\tilde{A}}(y) > \varphi_e(c_n)$. Now note that c_n was defined to be the stage where we last finished tempting within block B_n, and since there was no action in any B_m with $m < n$ after c_n was defined for the last time, it follows that $c_A(x) \leq c_n$ for all $x \in B_n$. Hence for $x \geq y$ in B_n, $C_{\tilde{A}}(x) = C_{\tilde{A}}(y) > \varphi_e(c_n) \geq \varphi_e(c_A(x))$.

Now for $x < y$ in B_n, we have $C_{\tilde{A}}(x) = C_{\tilde{A}}(\max\{B_{n-1}\})$ and $C_A(x) = C_A(\max\{B_{n-1}\})$. This is because we ensured that any action taken in the B_{n-1} block would effect the computation functions on values in the B_n block, and because there was action in the B_{n-1} block since φ_e is total. Since $e < n$, φ_e would have been considered in the B_{n-1} block, and so by an argument as above, we have $C_{\tilde{A}}(\max\{B_{n-1}\}) > \varphi_e(C_A(\max\{B_{n-1}\}))$. □

The lemma shows that if φ_e is total, then for almost all x, $C_{\tilde{A}}(x) > \varphi_e(C_A(x))$, completing the proof of the theorem.

□

Theorem 3.3. *For any $n \geq 3$ and any properly n-c.e. set D there exists an n-c.e. set $A \geq_m D$ such that A has two n-c.e. approximations $\{A_s\}$ and $\{\tilde{A}_s\}$, so that for any computable function f, C_A dominates $f \circ C_{\tilde{A}}$*

Proof. Modify the construction of the previous theorem, by placing in between the blocks "coding locations", and increasing the size of all blocks so that each block B_m will contain $(n+1)m$-many more sub-blocks. That is, ω will be arranged as $B_0 < d_0 < B_1 < d_1 < B_2 ...$. Proceed as in the construction of the previous theorem, except copy the set D into the coding locations, using its n-c.e. approximation. That is, have $x \in D_s \iff d_x \in A_s$ and $x \in D_s \iff d_x \in \tilde{A}_s$. Whenever some x enters or exits D, then in addition to causing d_x to enter or exit A and $\tilde{A}$, also reset c_m for all $m > x$, and shift the active sub-block down by one in B_m. Since D is n-c.e. , there can be only nm-many extra changes caused by the coding, so there will be enough sub-blocks to run the construction. □

References

[1] S. Barry Cooper. *Computability theory*. Chapman & Hall/CRC, Boca Raton, FL, 2004.

[2] Barbara F. Csima. Comparing c.e. sets based on their settling times. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *CiE*, volume 4497 of *Lecture Notes in Computer Science*, pages 196–204. Springer, 2007.

[3] Barbara F. Csima and Robert I. Soare. Computability results used in differential geometry. *J. Symbolic Logic*, 71(4):1394–1410, 2006.

[4] Robert I. Soare. *Recursively enumerable sets and degrees*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987. A study of computable functions and computably generated sets.

[5] Robert I. Soare. Computability theory and differential geometry. *Bull. Symbolic Logic*, 10(4):457–486, 2004.

[6] Robert I. Soare. *Computability Theory and Applications*. to appear.

Department of Pure Mathematics, University of Waterloo, Waterloo, ON, Canada N2L 3G1

URL: www.math.uwaterloo.ca/∼csima

E-mail address: csima@math.uwaterloo.ca