Auxiliary errata file for
***Algorithmic Randomness and Complexity*,**
by Rodney G. Downey and Denis R. Hirschfeldt

Last updated: July 5, 2012

Proof of Theorem 11.4.1. We build on the terminology and notation of the previous section, and introduce an improved version of the decanter method.

Let A be K-trivial. In approximating A', when we see a convergent computation $\Phi_e^{A_s}(e)$, we need to decide whether to believe this computation. If we make only finitely many mistakes for each e, then we ensure that A is low, while if we can keep the number of mistakes below a computable bound, then we ensure that A is superlow. There is no claim to uniformity in the theorem, so we can have infinitely many guessing procedures, as long as one them is successful.

As before, we exploit the fact that changes to the approximation of a K-trivial set come at a certain cost, in terms of additional short descriptions of initial segments. By loading measure beyond the use of a convergent computation $\Phi_e^{A_s}(e)$, we ensure that any violation of this use will come at a relatively large cost. Only once we have loaded sufficient measure will we believe this computation. This procedure ensures that the number of times we believe an incorrect computation of this form for a given e is computably bounded. (If the cost to the opponent of violating the use of a computation is made to be 2^{-k}, say, then we know we will believe no more than 2^k many incorrect computations.) Thus we can approximate A' in an ω-c.e. manner, whence A is superlow.

In the proof of Theorem 11.3.1, we built a c.e. set B and assumed we had a reduction $\Gamma^A = B$. Whenever we wanted to change A at a stage s to transform a $(j-1)$-set C_{j-1} into a j-set, we could do so simply by enumerating some number m_j into B, since we arranged things so that an A-change below $\gamma^A(m_j)[s]$ was sufficient to make C_{j-1} into a j-set, and such a change was guaranteed to happen once m_j entered B. What caused us problems was that A could change *before* we wanted it to, but we had no trouble ensuring that A would change once we wanted it to. When showing that A is superlow, rather than merely incomplete, we no longer have quite so much power. If the opponent is to prevent A from being superlow, it has to change A reasonably often, but we cannot force such changes to occur every time we would like them to.

We get around this problem by further distributing our measure-loading strategy among subprocedures. In the proof of Theorem 11.3.1, a procedure P_j called a P_{j-1} procedure several times, thus obtaining several $(j-1)$-sets. When it had enough such sets, it then promoted them to j-sets by forcing appropriate changes in A. The analogous procedure in this proof will run several subprocedures simultaneously. Each subprocedure will be associated with some index e. Each $(j-1)$-set C produced by such a subprocedure will have its corresponding measure loaded beyond some number n, so that a change of the approximation to $A \upharpoonright n$ to a hitherto unseen string will cause C to become a j-set. Our level j procedure will assume that such changes do not occur and, under that assumption, will guess at values of A'. That is, when it has an m such that a change in $A \upharpoonright m$ would allow it to transform a $(j-1)$-set C of sufficient weight given to it by a subprocedure associated with e into a j-set, and it sees that $\Phi_e^{A_s}(e)\downarrow$ with use below m, then it will believe that computation. If it is wrong, then C will be promoted. If it is wrong sufficiently often for subprocedures associated with some e, then it will build up a j-set with weight equal to its goal, and hence will return. If not, then it will be able to bound the number of wrong guesses it makes about the value of each $A'(e)$, and hence ensure that A is superlow. (For technical reasons, we will allow a procedure to return up to twice its goal.)

We envision the construction as happening on an infinitely branching tree of finite depth k (where k is as in the proof of Theorem 11.3.1), with each procedure on the tree having infinitely many subprocedures below it. As in the proof of Theorem 11.3.1, the top procedure will attempt to produce a k-set of inconsistently large measure, and hence will never return. Thus, at some point in the construction, we will start a run of some procedure that neither returns nor is canceled, but such that all the subprocedures it ever calls either return or are canceled. As outlined above, this run will succeed in guessing A' with sufficiently few mistakes to ensure that A is superlow. We call such a run a *golden run*.

It will be convenient to have two kinds of procedures. The procedure P_j calls subprocedures $Q_{j-1,e,u}$, corresponding to potential uses u of convergent computations $\Phi_e^{A_s}(e)$, which in turn call procedure P_{j-1} (except for the procedures $Q_{1,e,u}$, which are the ones that actually perform the measure loading by enumerating requests into a KC set L). Each call to a procedure comes with two parameters, as in the proof of Theorem 11.3.1, a goal g and a trash quota q. As in that proof, a run of a procedure may be canceled due to a premature A-change, in which case it will generate a certain amount of

"wasted measure". Again as in that proof, the trash quota will bound that amount. We choose our trash quotas to ensure that the total amount of wasted measure produced over the entire construction is small, and choose the goals of calls to subprocedures made by a particular run of a procedure to be small enough that the measure wasted by their potential cancelations adds up to no more than the trash quota of that run.

We now proceed with the formal details of the description of the procedures and the construction. The approximation to A' showing that A is superlow will not happen during the construction, but will be built after the fact, in the verification. Let b be a constant of K-triviality for A. We assume we have sped up our stages sufficiently so that $K_s(A \upharpoonright n) \leqslant K_s(n) + b$ for all s. Let d be a coding constant, given by the recursion theorem, for the machine corresponding to the KC set L we build. Let $k = 2^{b+d+1}$.

Procedure $P_i(g, q)$, where $1 < i \leqslant k$ and $g = 2^{-l} \geqslant q = 2^{-r}$ for some l and r. This procedure enumerates a set C, initially empty. At stage s, declare s to be *available*. (Availability is a local notion, applicable only to this particular run of this procedure.) For each $e \leqslant s$, proceed as follows.

1. If e is available and $\Phi_e^A(e)[s]\downarrow$, then let u be the use of this computation and call $Q_{i-1,e,u}(2^{-(e+2)}q, \min(2^{-(e+2)}q, 2^{-(2i+n)}))$, where n is the number of runs of Q_{i-1}-procedures that have been started by this point. Declare e to be unavailable.

2. If e is unavailable due to a run of $Q_{i-1,e,u}(g', q')$ and $A_s \upharpoonright u \neq A_{s-1} \upharpoonright u$, then declare e to be available, say that this run of $Q_{i-1,e,u}(g', q')$ is *released* and proceed as follows, where D is the set being built by the run of $Q_{i-1,e,u}(g', q')$.

   (a) Put D into C. If the weight of C (defined as in the proof of Theorem 11.3.1) is less than g then proceed to step 2(b). Otherwise, cancel all runs of subprocedures, and end this run of P_i, returning C. (Note that in this case the weight of C is less than $2g$.)

   (b) If the run of $Q_{i-1,e,u}(g', q')$ has not yet returned, then cancel it and all runs of subprocedures it has called.

Procedure $Q_{j,e,u}(g, q)$, where $1 < j < k$ and $g = 2^{-l} \geqslant q = 2^{-r}$ for some l and r. This procedure enumerates a set D, initially empty. Call $P_j(\frac{q}{2}, \min(\frac{q}{2}, 2^{-(2j+3+n)}))$, where n is the number of runs of P_j procedures

started so far. If this procedure returns a set C, then put C into D. If the weight of D is less than g then repeat this process, calling a new run of P_j. Otherwise, end the procedure, returning D. (It is easy to see that in this case the weight of D is less than $2g$.)

Procedure $Q_{1,e,u}(g, q)$*, where* $g = 2^{-l} \geqslant q = 2^{-r}$ *for some* l *and* r. Choose a fresh large number n and put (r, n) into L. Wait for a stage $t > n$ such that $K_t(n) \leqslant r + d$ and then put n into D. If the weight of D is less than g then repeat this process. Otherwise, end the procedure, returning D. (In this case the weight of D is equal to g.)

The construction begins by calling $P_{k,0}(1, 2^{-3})$. At each stage we perform one more cycle of each procedure that is currently running, descending through the levels and working in some effective order within each level.

We now verify that this construction works as intended. Let D_j be the union of the sets enumerated by Q_j-procedures. For $j > 1$, let C_j the union of the sets enumerated by P_j-procedures, and let $C_1 = \{n : \exists r\, ((r, n) \in L)\}$.

Lemma 11.4.2. *Each* C_i *is an* i*-set.*

Proof. For each $n \in D_1$, let s_n be the stage at which n enters D_1, and let r_n be the unique number such that $(r_n, n) \in L$. By construction, we have $K_{s_n}(A_{s_n} \upharpoonright n) \leqslant r_n + b + d$. Let $i \in [2, k]$ and assume by induction that for every $n \in D_{i-1}$, if n enters D_i at stage t then there are at least $i - 1$ many distinct strings σ of the form $K_v \upharpoonright n$ for $v \in [s_n, t]$ such that $K(\sigma) \leqslant r+b+d$. We show that C_i also has this property, which clearly implies that it is an i-set, and that D_i also has the property if $i < k$, so that the induction may continue. Suppose that n enters C_i at stage t. This enumeration happens at step 2(a) in the action of a run of some P_i-procedure for some e, and corresponds to the release of some run of a subprocedure $Q_{i-1,e,u}(g', q')$. By the inductive hypothesis, there are $i-1$ distinct strings σ of the form $A_v \upharpoonright n$ for $v \in [s_n, t)$ such that $K_v(\sigma) \leqslant r+b+d$. Furthermore, the approximation to $A \upharpoonright u$ cannot have changed between stages s_n and $t - 1$, since such a change would have caused the run of $Q_{i-1,e,u}(g', q')$ to have been released sooner. Since $A_t \upharpoonright u \neq A_{t-1} \upharpoonright u$, there is a new string $\tau = A_t \upharpoonright n$, not equal to any of these σ, such that $K_t(\tau) \leqslant r + b + d$. Since n is an arbitrary element of C_i, we see that C_i has the desired property, an in particular is an i-set. □

The next lemma shows that trash quotas are respected.

Lemma 11.4.3. (i) *For $j \in [1,k)$, the weight of the set of numbers in $C_j \setminus D_j$ corresponding to a run of $Q_{j,e,u}(g,q)$ is at most q.*

(ii) *For $i \in (1,k]$, the weight of the set of numbers in $D_{i-1} \setminus C_i$ corresponding to a run of $P_i(g,q)$ is at most q.*

Proof. (i) There is at most one n corresponding to a run of $Q_{1,e,u}(g,q)$ in $C_1 \setminus D_1$, since no number is put into C_1 by this run until the previous number it put into C_1 enters D_1. For such an n, the unique $(r,n) \in L$ is such that $2^{-r} = q$. Thus the weight of n is q.

If $j > 1$, then all numbers in $C_j \setminus D_j$ corresponding to a run of $Q_{j,e,u}(g,q)$ correspond to a single run of $P_j(\frac{q}{2}, q')$, because each time such a run returns, its numbers enter D_j. Thus this number never returns, and hence the weight of the set of numbers corresponding to it is at most q.

(ii) Let n be one of the numbers corresponding to a run of $P_i(g,q)$ that is in D_{i-1} by a stage t but never enters C_i. Then n is put in D_{i-1} by a run of a subprocedure $Q_{i-1,e,u}(2^{-(e+1)}q, q')$ called by this run of $P_i(g,q)$. It cannot be the case that $A_s \upharpoonright u \neq A_{s-1} \upharpoonright u$ for some $s > t$, as otherwise n would enter C_i at stage s. Thus the run of $Q_{i-1,e,u}$ that put n into D_{i-1} is never released, and e is never again available, so this run of $P_i(g,q)$ never calls a subprocedure $Q_{i-1,e,u'}$ after stage t.

Thus, for each e there is at most one run of a subprocedure of the form $Q_{i-1,e,u}(2^{-(e+2)}q, q')$ called by our run of $P_i(g,q)$ that leaves numbers in $D_{i-1} \setminus C_i$. Thus the sum of the weights of these numbers over all such e is at most $\sum_e 2^{-(e+1)}q = q$. □

Lemma 11.4.4. *L is a KC set.*

Proof. Since C_k is a k-set and $k = 2^{b+d+1}$, the weight of C_k is at most $\frac{1}{2}$, while the weight of $C_1 \setminus C_k$ is less than or equal to $\sum_{1 \leqslant j < k} C_j \setminus D_j + \sum_{1 < i \leqslant k} D_{i-1} \setminus C_i$, which by the previous lemma is bounded by the sum of the trash quotas of all procedures. These quotas were chosen so that this sum is at most $\frac{1}{2}$, so the weight of C_1 is at most 1. But the weight of L is equal to the weight of C_1, so L is a KC set. □

There cannot be a k-set of weight 1, so the top procedure started at the beginning of the construction can never return, and of course it is never canceled. Since there are only finitely many levels of procedures, there must be some run of some procedure that neither returns nor is canceled, but such that all the subprocedures it calls either return or are canceled. In other

words, there is a golden run of some procedure. This procedure cannot be a Q-procedure, since each run of a Q-procedure calls P-subprocedures with the same goal, and never cancels any of them itself, so that if it is never canceled, then it eventually fulfills its goal and returns. Thus the golden run is of some procedure P_i. We now show how to guess at A' in an ω-c.e. way.

Fix e. We begin by guessing that $e \notin A'$. Every time the golden run sees that $\Phi_e^A(e)[s]\downarrow$, it starts a run of a subprocedure $Q_{i-1,e,u}$, where u is the use of this computation. This run returns unless this use is violated before it can do so. If the subprocedure returns, we guess that $e \in A'$. If the use u is later violated, then we go back to guessing that $e \notin A'$. Every time a run of a subprocedure $Q_{i-1,e,u}$ returns, it adds at most $2^{-(e+1)}2$ to the weight of the set C associated with the golden run. Hence, we cannot guess wrongly at whether e is in A' more than $\frac{2^e g}{q}$ many times, lest the golden run reach its goal and return. Thus A is superlow. □

Proof of Theorem 11.4.9. The proof is essentially the same as that of Theorem 11.4.1, except that the "triggering condition" for having a P-strategy launch a Q-substrategy changes. Thus we limit ourselves to giving the description of the procedures (which will be quite similar to the ones in the proof of Theorem 11.4.1) and the necessary changes to the verification.

Procedure $P_i(g, q)$, where $1 < i \leqslant k$ and $g = 2^{-l} \geqslant q = 2^{-r}$ for some l and r. This procedure enumerates a set C, initially empty. At stage s, declare each $\sigma \in 2^s$ to be *available*. For each $\sigma \in 2^{\leqslant s}$, proceed as follows.

1. If σ is available and $\mathcal{U}^A(\sigma)[s]\downarrow = \tau$ for some τ, then let u be the use of this computation and call $Q_{i-1,\sigma,\tau,u}(2^{-|\sigma|+1}q, \min(2^{-|\sigma|+1}q, 2^{-(2i+n)}))$, where n is the number of runs of Q_{i-1}-procedures that have been started by this point. Declare σ to be unavailable.

2. If σ is unavailable due to a run of $Q_{i-1,\sigma,\tau,u}(g', q')$ and $A_s \upharpoonright u \neq A_{s-1} \upharpoonright u$, then declare σ to be available, say that this run of $Q_{i-1,\sigma,\tau,u}(g', q')$ is *released* and proceed as follows, where D is the set being built by the run of $Q_{i-1,\sigma,\tau,u}(g', q')$.

   (a) Put D into C. If the weight of C is less than g then proceed to step 2(b). Otherwise, cancel all runs of subprocedures, and end this run of P_i, returning C.

   (b) If the run of $Q_{i-1,\sigma,\tau}(g', q')$ has not yet returned, then cancel it and all runs of subprocedures it has called.

Procedure $Q_{j,\sigma,\tau,u}(g,q)$*, where* $1 < j < k$ *and* $g = 2^{-l} \geqslant q = 2^{-r}$ *for some* l *and* r. This procedure enumerates a set D, initially empty. Call $P_j(\frac{g}{2}, \min(\frac{g}{2}, 2^{(-2j+3+n)}))$, where n is the number of runs of P_j procedures started so far. If this procedure returns a set C, then put C into D. If the weight of D is less than g then repeat this process, calling a new run of P_j. Otherwise, end the procedure, returning D.

Procedure $Q_{1,\sigma,\tau,u}(g,q)$*, where* $g = 2^{-l} \geqslant q = 2^{-r}$ *for some* l *and* r. Choose a fresh large number n and put (r,n) into L. Wait for a stage $t > n$ such that $K_t(n) \leqslant r + d$ and then put n into D. If the weight of D is less than g then repeat this process. Otherwise, end the procedure, returning D.

The construction runs as before. The verification that there is a golden run is also essentially as before, except for the last paragraph of the proof of Lemma 11.4.3. In this case, the same argument as in that proof shows that for each σ, there is at most one run of a subprocedure of the form $Q_{i-1,\sigma,\tau,u}(2^{-|\sigma|+1}q, q')$ called by our run of $P_i(g,q)$ that leaves numbers in $D_{i-1} \setminus C_i$. Furthermore, none of these subprocedures are ever released, so all of the corresponding σ's are in $\mathrm{dom}\mathcal{U}^A$, and thus the sum of the weights of these numbers over all such σ is bounded by $\Omega^A q < q$.

Thus, all that is left to do is to take a golden run of some $P_i(g,q)$ and use it to show that A is low for K. To do so, we build a KC set W. Let $c = \log \frac{g}{q}$. Whenever a run of $Q_{i-1,\sigma,\tau,u}(2^{-|\sigma|+1}q, q')$ called by the golden run returns, put $(|\sigma| + c + 1, \tau)$ into W. We first show that W is a KC set.

Suppose that $(|\sigma| + c + 1, \tau)$ enters W at stage s because a run of $Q_{i-1,\sigma,\tau,u}(2^{-|\sigma|+1}q, q')$ returns, and $A_t \upharpoonright u = A_s \upharpoonright u$ for all $t > s$. Then $\sigma \in \mathrm{dom}\mathcal{U}^A$, so the weight contributed by all such requests is bounded by $2^{-(c+1)}\Omega^A < \frac{1}{2}$.

Now suppose that $(|\sigma| + c + 1, \tau)$ enters W at stage s because a run of $Q_{i-1,\sigma,\tau,u}(2^{-|\sigma|}q, q')$ returns, and there is a $t > s$ such that $A_t \upharpoonright u \neq A_s \upharpoonright u$. Then the set D returned by this run has weight at most $2^{-|\sigma|}q$ and enters the set C built by the golden run. Thus the weight contributed by all such requests is bounded by $\frac{2^{-(c+1)}}{q}$ times the weight of C. Since $\frac{2^{-(c+1)}}{q} = \frac{1}{2g}$ and the weight of C is bounded by g, the weight contributed by all such requests is bounded by $\frac{1}{2}$.

Now fix τ and let s be the least stage such that for some σ of length $K^A(\tau)$, we have $\mathcal{U}^A(\sigma)\downarrow[s] = \tau$ with use u and $A_t \upharpoonright u = A_s \upharpoonright u$ for all $t > s$. It follows easily from the minimality of s that σ must be available at s. Thus a run of $Q_{i-1,\sigma,\tau,u}$ is launched. This run is never canceled, and thus must

return, by the definition of golden run. So $(|\sigma| + c + 1, \tau) \in W$, and hence $K(\tau) \leqslant K^A(\tau) + O(1)$, where the constant does not depend on τ. Since τ is arbitrary, A is low for K. $\square$