Traceable sets

Rupert Hölzl and Wolfgang Merkle*

Institut für Informatik,
Ruprecht-Karls-Universität,
Heidelberg, Germany,
{hoelzl|merkle}@math.uni-heidelberg.de

Abstract. We investigate systematically into the various possible notions of traceable sets and the relations they bear to each other and to other notions such as diagonally noncomputable sets or complex and autocomplex sets. We review known notions and results that appear in the literature in different contexts, put them into perspective and provide simplified or at least more direct proofs. In addition, we introduce notions of traceability and complexity such as infinitely often versions of jump traceability and of complexity, and derive results about these notions that partially can be viewed as a natural completion of the previously known results. Finally, we give a result about polynomial-time bounded notions of traceability and complexity that shows that in principle the equivalences derived so far can be transferred to the time-bounded setting.

1 Introduction and overview

The various notions of a traceable set have received quite a lot of attention in the area of algorithmic randomness. On the one hand, traceability naturally comes up in connection with lowness notions, as it is exemplified in the work of Terwijn and Zambella [12] on Schnorr randomness and, more recently, the attempts to characterize lowness for Martin-Löf randomness and the equivalent notion of K-triviality by an appropriate version of jump traceability [1, 3]. On the other hand, traceability has been shown [9] to interact informatively with classical notions from computability theory such as diagonally noncomputable sets and with notions such as autocomplex that are defined in terms of Kolmogorov complexity of initial segments of sets.

In this article, we investigate into notions of traceability from a systematic point of view. We review standard notions of traceability and some basic results known on them, giving simplified or at least more direct proofs than in the current literature, which in particular are meant to provide an intuitive picture of why the stated relations hold. One of our aims is to give a unified view of notions and results that appear in the literature, and for example we argue that a recent results on anticomplex sets by Franklin et al. [5] can be seen as a variant of results on the relations between notions of complexity and i.o. traceability [9].

* The first and the second author are supported and partially supported, respectively, by DFG grant ME 1806/3-1.

We also introduce new notions of traceability such as infinitely often versions of jump traceability and derive an interesting collapse result. Finally, we give a result about polynomial-time bounded notions of traceability and complexity that shows that in principle the equivalences derived so far can be transferred to the time-bounded setting.

Notation In the sequel, set refers to a subset of the natural numbers $\mathbb{N}$ and functions and partial functions map natural numbers to natural numbers, unless explicitly specified differently. We let $\mathrm{W}_0, \mathrm{W}_1, \ldots$ be the standard acceptable numbering of all computably enumerable (c.e.) sets, i.e., W_e is the domain of the e-th partial computable function φ_e. Let C and K denote the plain and prefix-free versions of Kolmogorov complexity [4, 10]. Let $\leq^+$ denote the relation less than or equal to up to an additive constant, and $\geq^+$ is defined likewise.

2 Traceability

The various traceability notions considered in the sequel are either well-known or have at least been considered implicitly in the literature, except for, to the best of our knowledge, the infinitely often versions of jump traceable and strongly jump traceable introduced in Definition 9 below.

Definition 1. *A* trace *is a sequence $(T_n)_n$ of sets. A trace $(T_n)_n$* is a trace for *a partial function f if $f(n) \in T_n$ holds for all n such that $f(n)$ is defined. A trace $(T_n)_n$ is an* i.o. trace for *a partial function f if there are infinitely many n such that $f(n) \in T_n$.*

We will also say, for short, that a trace traces or i.o. traces a partial function f, in case the trace is a trace or an i.o. trace, respectively, for f. For the traces $(T_n)_n$ considered in the sequel, the sets T_n will always be finite.

Definition 2. *For a function h, a trace $(T_n)_n$ is* h-bounded, *if $|T_n| \leq h(n)$ holds for all n.*

A trace $(T_n)_n$ is computably enumerable (c.e.) *if there is a computable function g such that T_n is equal to $\mathrm{W}_{g(n)}$ for all n. A trace $(T_n)_n$ is* computable *if there is a computable function g such that T_n is equal to $\mathrm{D}_{g(n)}$ for all n, where D_e is the finite set with canonical index e.*

Definition 3. *An* order *is a nondecreasing and unbounded function. A set A is* c.e. traceable *iff there is a computable order h such that all functions $f \leq_{\mathrm{T}} A$ are traced by an h-bounded c.e. trace $(T_n)_n$. A set A is* c.e. i.o. traceable *iff there is a computable order h such that all functions $f \leq_{\mathrm{T}} A$ are i.o. traced by an h-bounded c.e. trace $(T_n)_n$.*

The concepts of computably traceable *and of* computably i.o. traceable *are defined similarly where in addition the traces are required to be computable instead of being merely c.e.*

For all the concepts introduced above, there are variants where Turing reducibility is replaced by weak truth-table or truth-table reducibility, e.g., we say a set A is i.o. wtt-traceable *iff there is a computable order h such that all functions $f \leq_{\mathrm{wtt}} A$ are i.o. traced by an h-bounded c.e. trace $(T_n)_n$.*

Remark 4. Stephan [11] made the interesting observation that a set is c.e. traceable if and only if there is a computable function h such that every $f \leq_T A$ satisfies $\mathrm{C}(f(n)) < h(n)$ for almost all n. This characterization has the advantage that it works without defining traces and just uses classical concepts. The disadvantage of this style of characterization is that for other traceability concepts it yields more complicated equivalences; for example the case of computable traceability would require the use of Kolmogorov complexity defined over total machines.

Terwijn and Zambella [12] observed that the notions of computable and c.e. traceability remain the same if one requires in their respective definitions the existence of h-bounded traces not just for a single but for all computable orders h. The corresponding argument extends directly to the notions c.e. and computably wtt-traceable, as well as c.e. and computably tt-traceable, but also to the infinitely often versions of these notions, as is shown in the following remark. For the notion of i.o. c.e. traceable this also follows by Theorem 10 below, and, what is more, by Corollaries 21 and 23 for some notions even the existence of 1-bounded traces of the considered type is equivalent.

Remark 5. A set A is c.e. i.o. traceable if and only if for all computable orders h all functions $f \leq_{\mathrm{T}} A$ are i.o. traced by an h-bounded c.e. trace $(T_n)_n$, and a similar statement holds for the notion computably i.o. traceable, as well as for variants of these notions defined in terms of weak truth-table or truth-table reducibility in place of Turing reducibility.

The proof uses the same technique as the proof [12] for the analogous everywhere version of the statement. Let us assume we have i.o. traces bounded by a computable order g and let us construct a i.o. trace $(S_n)_n$ for some function $f \leq_T A$ bounded by some given computable order h.

Let $\hat{g}(n)$ be the least number k such that $h(k) \geq g(n)$. This is computable and well-defined. Therefore the mapping $\hat{f}$ defined by $i \mapsto (f(0), \ldots, f(\hat{g}(i+1)))$ is Turing-reducible to A and therefore has a trace $(T_i)_i$ with bound g.

Let $\hat{g}^{-1}$ be the discrete inverse of $\hat{g}$, that is, for a given k, $\hat{g}^{-1}(k)$ is the largest number n such that $g(n) \leq h(k)$. Then define $(S_n)_n$ by

$$S_n := \{\pi_n(x) \colon x \in T_{\hat{g}^{-1}(n)}\}$$

where π_n is the projection to the n-th coordinate.

Then S_n has at most $g(\hat{g}^{-1}(n)) \leq h(n)$ entries. For infinitely many i, T_i is right; that is, it contains some correct $\hat{g}(i+1)$-tuple $(f(0), \ldots, f(\hat{g}(i+1)))$. This tuple then contains (among other) the correct information about the values of all $f(n)$ with n such that $\hat{g}^{-1}(n) = i$. So S_n will be a correct trace for $f(n)$ for all such n. □

The following theorem is attributed to Kjos-Hanssen et al. [9] by Downey and Hirschfeldt [4], however, the assertion of the theorem does not even implicitly appear in the published versions of the corresponding article [9], nor does its proof. Since the proof presented by Downey and Hirschfeldt is via a chain of

equivalent statements, we consider it useful and instructive to give a direct argument here. Among the various equivalent definitions for the notion high, we will work with the one according to which a set A is high iff A computes a function that dominates every computable function.

Theorem 6. *The following statements are equivalent.*

(i) The set A is computably i.o. traceable.
(ii) The set A is c.e. i.o. traceable and nonhigh.

Proof. (i) implies (ii): Any computably i.o. traceable set A is *a fortiori* c.e. i.o. traceable, and is also nonhigh because given an A-computable function g we obtain a computable function f such that $g(n) \leq f(n)$ for infinitely many n by letting $f(n) = 1 + \max T_n$ where $(T_n)_n$ is a computable trace for g.

(ii) implies (i): Let us assume we have a i.o. trace $(T_n)_n$ of a function $\ell \leq_T A$. Define the function g such that on argument n one starts to enumerate in parallel the traces T_m for all $m \geq n$ and A-computably recognizes when for the first time for some m_n the correct value $\ell(m_n)$ is enumerated into T_{m_n}, then letting $g(n)$ be the number of computational steps of the enumeration of T_{m_n} that are required to enumerate $\ell(m_n)$. In this situation, let us say that n has found m_n. Since g is computable in A and A is nonhigh, there is a computable function f that at infinitely many places is larger than g, where in addition we can assume that f is nondecreasing.

We can now get a computable trace $(\widetilde{T}_n)_n$ for ℓ that is correct at infinitely many places as follows: simply let $\widetilde{T}_n$ contain all elements that are enumerated into T_n in at most $f(n)$ steps.

This trace is correct infinitely often. Indeed, *any* n finds *some* m_n, and among the corresponding pairs (n, m_n) there are infinitely many where we have

$$g(n) \leq f(n) \leq f(m_n),$$

i.e., for these pairs $f(m_n)$ exceeds the number of steps needed to enumerate $\ell(m_n)$ into T_{m_n}, so for these pairs the the correct value $\ell(m_n)$ will be a member of $\widetilde{T}_{m_n}$.

Finally observe that in the construction the set $\widetilde{T}_n$ is always contained in T_n, hence any uniform bound h for the c.e. traces of the functions computable in A will also be a uniform bound for the corresponding computable traces. □

We review the concepts of jump traceable and strongly jump traceable, which can be seen as stricter versions of the notion of c.e. traceable where not only the total but also all partial functions computable in a given set must be traced.

Definition 7. *A set A set is* jump traceable *iff there is a computable order h such that for all functions partially computable in A there is an h-bounded c.e. trace. A set A is* strongly jump-traceable *iff for all computable orders h it holds that for all functions partially computable in A there is an h-bounded c.e. trace.*

Remark 8. It is easier for our purposes to work with the given definition. Alternatively, jump traceability can be defined by requiring that the diagonal jump function is traceable. For more details, see Downey and Hirschfeldt [4].

It is well-known that the class of strongly jump-traceable sets is a proper subclass of the jump-traceable sets, in fact, the two classes are proper sub- and superclasses, respectively, of the class of K-trivial sets [1, 3]. However, for the infinitely often versions of these two notions we get an interesting collapse of traceability notions.

Definition 9. *A set A is* i.o. jump-traceable *iff there is a computable order h such that for all functions partially computable in A that have an infinite domain there is an h-bounded c.e. i.o. trace.*

A set A is strongly i.o. jump-traceable *iff for all computable orders h it holds that for all all functions f partially computable in A that have an infinite domain there is an h-bounded c.e. i.o. trace.*

Theorem 10. *The following statements are equivalent.*

(i) The set A is strongly i.o. jump-traceable.
(ii) The set A is i.o. jump-traceable.
(iii) The set A is c.e. i.o. traceable.

Proof. By definition, (i) implies (ii) and (ii) implies (iii), so it suffices to show that not strongly i.o. jump traceable implies not c.e. i.o. traceable. So let A be a set that computes a partial function f that for some computable order h_0 cannot be i.o. traced by any h_0-bounded c.e. trace. We show that for any given computable order h there is an A-computable function that cannot be i.o. traced by any h-bounded c.e. trace. Fix an appropriate effective enumeration $(T^0_n)_n, (T^1_n)_n, \ldots$ of all h-bounded c.e. traces, e.g., let T^e_n be the subset of the n-th row of W_e that contains the first $h(n)$ elements that are enumerated into this row. Furthermore, let S_n be the union of all T^e_i where $i < n$ and $e < n$ and observe that this way the cardinality of S_n is at most $c(n) = n^2 h(n)$. For all n, let T_n be equal to S_m where m is maximum such that $c(m) \leq h_0(n)$ and call the trace $(T_n)_n$ the universal h_0-bounded trace, which by construction is indeed h_0-bounded, hence does not i.o. trace f. Hence for almost all m such that $f(m)$ is defined, we have $f(m) \notin T_m$. So we obtain an A-computable function as required by mapping n to a value of the form $f(m)$ such that this value is defined and $c(n) \leq h_0(m)$. □

In order to render the statement of results in Section 5 and 6 more intuitive, we introduce the following alternate notation for notions of not being traceable.

Definition 11. *A set* avoids c.e. traces *if the set is not c.e. i.o. traceable and the set* i.o. avoids c.e. traces *if it is not c.e. traceable. Similarly, a set* tt-avoids c.e. traces *if the set is not c.e. i.o. tt-traceable, and further notions such as c.e. wtt-avoiding computable traces are defined in the same manner.*

3 Autocomplex and complex sets

The notions of complexity and autocomplexity were first defined in an article by Kanovich [8], where he showed that autocomplex sets are Turing complete and complex sets are wtt-complete for the class of c.e. sets.

Definition 12. *A set A is* complex *if there is a computable order h such that for all n, it holds that $C(A \upharpoonright n) \geq h(n)$.*

A set A is called autocomplex if there is an A-computable order h such that for all n, it holds that $C(A \upharpoonright n) \geq h(n)$.

We omit the straightforward proof of the following known fact [4, 9]. Note that by the standard proof of Proposition 13 it is immediate that all the functions g that occur in the proposition can be assumed to be order functions.

Proposition 13. *A set A is complex if and only if there is a computable function g such that for all n, we have $C(A \upharpoonright g(n)) \geq n$ if and only if there is a function $g \leq_{\mathrm{tt}} A$ such that for all n, we have $C(g(n)) \geq n$ if and only if there is a function $g \leq_{\mathrm{wtt}} A$ such that for all n, we have $C(g(n)) \geq n$.*

A set A is autocomplex if and only if there is an A-computable function g such that for all n, we have $C(A \upharpoonright g(n)) \geq n$ if and only if there is an A-computable function g such that for all n, we have $C(g(n)) \geq n$.

In Section 6, we will see that it is interesting to consider variants of the notions autocomplex and complex where the condition $C(A \upharpoonright g(n)) \geq n$ is not required for all but just for infinitely many n. In connection with the following definition, note that the notion of *not* being i.o. complex has been introduced by Franklin et al. [5] under the name of anticomplex.

Definition 14. *A set A is* i.o. complex *iff there is a computable order g such that for infinitely many n, we have $\mathrm{C}(A \upharpoonright g(n)) \geq n$.*

A set A is i.o. autocomplex *iff there is an A-computable order g such that for infinitely many n, we have $\mathrm{C}(A \upharpoonright g(n)) \geq n$.*

The equivalent characterizations of complex suggest different ways to define i.o. complex (and similar remarks can be made for the notion i.o. autocomplex). However, it would neither be equivalent nor even make sense to define i.o. complexity by requiring that there is some computable order h such that for infinitely many n it holds that $\mathrm{C}(A \upharpoonright n) \geq h(n)$, because for small h such as the map $n \mapsto \log \log n$ this inequality is satisfied for infinitely many initial segments of any set A, simply because a code for $A \upharpoonright n$ is always also a code for n. In Section 8, we will see that equivalent definitions in this style are still possible by considering specific variants of Kolmogorov complexity. Furthermore, the two following propositions show that in the defining condition $\mathrm{C}(A \upharpoonright g(n)) \geq n$ of i.o. autocomplexity and i.o. complexity the lower bound n can equivalently be replaced by a wide range of lower bounds in case g may depend on this bound.

Proposition 15. *The following assertions are equivalent.*

(i) *The set A is i.o. autocomplex.*
(ii) *There is a computable order h and an A-computable function g such that there are infinitely many n where $\mathrm{C}(A \upharpoonright g(n)) \geq \mathrm{C}(n) + h(n)$.*
(iii) *For every A-computable order h there is an A-computable function g such that there are infinitely many n where $\mathrm{C}(A \upharpoonright g(n)) \geq h(n)$.*

Proof. It is immediate that (i) implies (ii) and that (iii) implies (i). For a proof of the remaining implication from (ii) to (iii), fix h and g that satisfy (ii), and let h_A be any A-computable order. Let $m_0 = 0$ and for all $n > 0$ let

$$m_n = \min\{m\colon m_{n-1} < m \text{ and } 3h_A(n) \leq h(m)\} \quad \text{and} \quad I_n = [m_n, m_{n+1}) \,.$$

For all n, let $\widetilde{g}(n)$ an appropriate representation of the pair of the restriction of g to I_n and the initial segment of A of length $\max_{j\in I_n} g(j)$, and observe that the function $\widetilde{g}$ is A-computable. By assumption on g and by construction, there are infinitely many j such that for the index n where $j \in I_n$, we have

$$\mathrm{C}(A \upharpoonright g(j)) \geq \mathrm{C}(j) + h(j) \geq \mathrm{C}(j) + h(m_n) \geq \mathrm{C}(j) + 3h_A(n) \,.$$

For each such j and n, it holds that $\mathrm{C}(\widetilde{g}(n)) \geq h_A(n)$, because otherwise $A \upharpoonright g(j)$ could be described by a word of length $\mathrm{C}(j) + 2h_A(n) + \mathrm{O}(1)$. □

The following variant of Proposition 15 can be shown by an almost literally identical proof, which we omit.

Proposition 16. *The following assertions are equivalent.*

(i) The set A is i.o. complex.
(ii) There is a computable order h and a computable function g such that there are infinitely many n where $\mathrm{C}(A \upharpoonright g(n)) \geq \mathrm{C}(n) + h(n)$.
(iii) For every computable order h there is a computable function g such that there are infinitely many n where $\mathrm{C}(A \upharpoonright g(n)) \geq h(n)$.

4 Diagonally noncomputable sets

Definition 17. *A set A is* diagonally noncomputable (DNC) *if there is a function $f \leq_{\mathrm{T}} A$ such that $f(n)$ differs from $\varphi_n(n)$ whenever the latter value is defined. With an appropriate coding scheme for finite sequences of natural numbers understood, a set A is* strongly diagonally noncomputable (SDNC) *if there is a function $f \leq_{\mathrm{T}} A$ such that when z is a code for the sequence $e_1, x_1, \ldots, e_m, x_m$, then $f(z)$ differs for $i = 1, \ldots, m$ from $\varphi_{e_i}(x_i)$ whenever this value is defined.*

The notions of wtt-DNC, wtt-SDNC, tt-DNC, *and* tt-SDNC *are defined likewise, where in the above definitions $f \leq_{\mathrm{T}} A$ is replaced by $f \leq_{\mathrm{wtt}} A$ and $f \leq_{\mathrm{tt}} A$, respectively.*

Note that if we can compute a function f such that for given n the value $f(n)$ differs from $\varphi_n(n)$, we can also compute a function g such that for given e, x the value $g(e, x)$ differs from $\varphi_e(x)$, because by the s-m-n theorem one can effectively find an index i such that $\varphi_e(x)$ and $\varphi_i(i)$ are either both undefined or both defined and have the same value. By a result of Jokusch [7], indeed even the notions of DNC and SDNC coincide.

Theorem 18. *A set A is DNC if and only if A is SDNC.*

Proof. By definition, it suffices to show that DNC implies SDNC. If A is DNC, one obtains an A-computable function f as required as follows. By fixing uniformly effective and uniformly effectively invertible bijections between $\mathbb{N}$ and $\mathbb{N}^m$, for any m, natural numbers can be uniquely identified with m-tuples of natural numbers. Then given a sequence $e_1, x_1, \ldots, e_m, x_m$ with code z, let $f(z)$ be equal to the m-tuple $(y_1, \ldots, y_m)$, where y_i differs from the i-th component of $\varphi_{e_i}(x_i)$, whenever this value is defined. □

The following infinitely often versions of the notion DNC is due to Kjos Hanssen et al. [9]. Note that there are computable functions g such that $g(e)$ differs from $\varphi_e(e)$ for infinitely many e, hence in order to get interesting infinitely often versions of the various variants of the concept of DNC, one has to require more than just to be able to compute a function that differs from the partial diagonal function at infinitely many places.

Definition 19. *For a function g, let $E_g = \{e\colon g(e) = \varphi_e(e)\}$ be the* (diagonal) equality set *of g. A set A is* i.o. DNC *if for all computable functions z there is a function $g \leq_{\mathrm{T}} A$ such that there are infinitely many n where*

$$|E_g \cap \{0, \ldots, z(n) - 1\}| \leq n \, .$$

The concepts of i.o. tt-DNC *and* i.o. wtt-DNC *are defined likewise, where in the definition $g \leq_{\mathrm{T}} A$ is replaced by $g \leq_{\mathrm{tt}} A$ and $g \leq_{\mathrm{wtt}} A$, respectively.*

By definition, a set A is DNC if and only if there is an A-computable function such that E_g is empty, and consequently any set that is DNC is also i.o. DNC. More precisely, if a set A is DNC, then it satisfies the definition of i.o. DNC by a function $g \leq_{\mathrm{T}} A$ that does not depend on z. It can be shown that the latter also holds true for a set that is i.o. DNC and high, and that a DNC set A is high if and only if there is a single function $g \leq_{\mathrm{T}} A$ that works for all z such that in addition E_g is infinite.

5 Equivalences of the almost everwhere notions

The following theorem is due to Kjos-Hanssen, Merkle and Stephan [9, Theorems 2.3 and 2.7]. The proof of their result given here is somewhat more direct, furthermore, their short but slightly technical proof of the implication from DNC to autocomplex is replaced by a simplified argument due to Khodyrev and Shen, who rediscovered the known equivalence of DNC and SDNC and observed that SDNC easily implies autocomplex. The equivalence results of this and the following sections are formulated in terms of avoidance as introduced in Definition 11 in order to render these results more intuitive.

Theorem 20. *The following assertions are equivalent.*

(i) The set A is autocomplex.
(ii) The set A is DNC.
(iii) The set A avoids c.e. traces.

Proof. First, assume that A is autocomplex. Then there is an A-computable function g such that for all n, we have $\mathrm{C}(g(n)) \geq n$. So $g(n)$ differs from $\varphi_n(n)$ for almost all n, because the latter value, if defined, has plain complexity of $\log n$ up to an additive constant, and consequently, A is DNC. Similarly, the set A is not c.e. i.o. traceable, i.e., avoids c.e. traces, because otherwise the function g had an n-bounded c.e. trace by Remark 5, which implied $\mathrm{C}(g(n)) \leq^+ 2\log n$.

Next assume that A is DNC and hence SDNC. Then A is autocomplex because in order to obtain for given n a value $g(n)$ where $\mathrm{C}(g(n)) \geq n$, it suffices to obtain a value that differs from all the values $\varphi_e(p)$ where the latter value is defined, p has length at most n, and e is an index for the universal machine used in the definition of the plain complexity C.

Finally, assume that the set A avoids c.e. traces, i.e., is not c.e. i.o. traceable. In order to see that A is DNC, let the diagonal trace $(T_n)_n$ be defined by $T(n) = \{\varphi_e(e)\}$. By assumption, there is an A-computable function g that is not i.o. traced by the diagonal trace, hence $g(e)$ differs from $\varphi_e(e)$, whenever the latter value is defined. □

Corollary 21. *A set A is c.e. i.o. traceable if and only if every A-computable function has a 1-bounded c.e. i.o. trace.*

Proof. It suffices to show the implication from left to right. By the proof of the implication from (iii) to (ii) in Theorem 20, if there is an A-computable function that has no 1-bounded c.e. i.o. trace, then this function witnesses that A is DNC, hence, by the same theorem, A is not i.o. c.e. traceable. □

The following variant of Theorem 20 is again due to Kjos-Hanssen et al. [9]. The proofs of Theorem 22 and its corollary are omitted because they are almost literally the same as for Theorem 22 and Corollary 21 when using the characterizations of the notion complex from Proposition 13 and showing separately the equivalences for truth-table and weak truth-table reducibility.

Theorem 22. *The following assertions are equivalent.*

(i) The set A is complex.
(ii) The set A is tt-DNC.
(iii) The set A tt-avoids c.e. traces.

The three assertions remain equivalent if one replaces in the two last assertions truth-table reducibility by weak truth-table reducibility.

Corollary 23. *The following assertions are all equivalent to A not being complex.*

(i) The set A is c.e. i.o. tt-traceable.
(ii) Every function $f \leq_{\mathrm{tt}} A$ has a 1-bounded c.e. i.o. trace.
(iii) The set A is c.e. i.o. wtt-traceable.
(iv) Every function $f \leq_{\mathrm{wtt}} A$ has a 1-bounded c.e. i.o. trace.

6 Equivalence of the infinitely often notions

In Section 5 we have seen equivalences between first, notions of complexity and autocomplexity, second, computing diagonally noncomputable functions, and third, notions of avoiding c.e. traces. The corresponding proofs were rather direct and functions g as required in the definitions of these three notions where obtained place by place in the sense that, for example, a function value $g(n)$ that has a certain complexity is obtained by considering a value $g(n)$ that is not contained in a component T_n of an appropriate trace and vice versa. Accordingly, by identical or similar arguments, we obtain infinitely often versions of these equivalence results where now, for example, for all n such that the value $g(n)$ has high complexity the value $g(n)$ avoids a corresponding set T_n and vice versa.

The two following theorems are infinitely often versions of Theorems 20 and 22. The equivalence of assertions (i) and (iii) in Theorem 25 for the case of weak truth-table reducibility is due to Franklin et al. [5].

Theorem 24. *The following assertions are equivalent.*

(i) The set A is i.o autocomplex.
(ii) The set A is i.o. DNC.
(iii) The set A i.o. avoids c.e. traces.

Proof. We first show that (i) and (iii) are equivalent, which follows by essentially the same arguments as the equivalence of being autocomplex and being DNC stated in Theorem 20. If A is i.o. autocomplex, then there is an A-computable function g such that for infinitely many n it holds that $\mathrm{C}(g(n)) \geq n$, and such a function g cannot have a c.e. trace that, e.g., is n-bounded, hence A is not c.e. traceable, i.e., A i.o. avoids c.e. traces. Conversely, if A i.o. avoids c.e. traces, there is an A-computable function g that has no 2^n-bounded c.e. trace, hence in particular, there are infinitely many n such that there is no word w of length strictly less than n such that $g(n) = U(w)$, where U is the universal machine used in the definition of C, and consequently A is i.o. autocomplex.

In order to show that (i) implies (ii), assume that A is i.o. autocomplex. Fix any computable function z and let $m_0, m_1, \ldots$ be a strictly increasing computable sequence of natural numbers such that for all i, we have $z(m_i) < m_{i+1}$. This way the natural numbers are partitioned into consecutive intervals $I_i = [m_i, m_{i+1})$. By Proposition 15, choose some A-computable function g_0 such that there are infinitely many n such that $\mathrm{C}(g_0(n)) \geq \max I_n$. For all n and all j in I_n, let $g(j) = g_0(n)$. Then g is A-computable and there are infinitely many n where for all j in I_n we have

$$\mathrm{C}(\varphi_j(j)) \leq^+ \log j < j \leq \max I_n \leq \mathrm{C}(g_0(n)) = \mathrm{C}(g(j)) \,,$$

i.e., the set E_g has an empty intersection with I_n and thus contains at most $m_n = \min I_n$ numbers that are less than or equal to $z(m_n) \leq \max I_n$.

In order to demonstrate that (ii) implies (iii), we show the contrapositive, so assume that A does not i.o. avoid c.e. traces, i.e., that A is c.e. traceable. Fix

some appropriate effective way of coding finite sequences of natural numbers of arbitrary length by single natural numbers. Let $(T^0_\ell)_{\ell\in\mathbb{N}}, (T^1_\ell)_{\ell\in\mathbb{N}}, \ldots$ be an appropriate effective enumeration of all c.e. traces. Let s be a computable function such that for all i and j the partial computable function $\varphi_{s(i,j)}$ on input y is computed by enumerating the numbers $c_0, c_1, \ldots$ in T^i_y until c_j is reached, where c_j is then considered as a code for a finite sequence of the form $g(0), g(1), \ldots, g(\ell)$ and in case $y \leq \ell$ the output is $g(y)$.

Next define a computable function z where for all n the value $z(n)$ is chosen so large that for all $i < n$ and $j < n$ there are at least $n+1$ mutually distinct indices $e \leq z(n)$ such that the partial function φ_e is the same as $\varphi_{s(i,j)}$. Then given any function $g \leq_{\mathrm{T}} A$, let $\widetilde{g}(n)$ be a code for the finite sequence $g(0), \ldots g(z(n))$. By assumption on A, for $h\colon n \mapsto n$ there is an index i such that the c.e. trace $(T^i_\ell)_{\ell\in\mathbb{N}}$ is h-bounded and traces the function $\widetilde{g}$. For given n, let j be minimum such that $\widetilde{g}(n) = c_j$, where $c_0, c_1, \ldots$ are the numbers that are enumerated into T^i_n. Then for all sufficiently large n, there are at least $n+1$ places $e \leq z(n)$ such that

$$\varphi_e(e) = \varphi_{s(i,j)}(e) = g(e)\,,$$

and since g was an arbitrary A-computable function and z does not depend on g, the set A is not i.o. DNC. □

Theorem 25. *The following assertions are equivalent.*

(i) The set A is i.o. complex.
(ii) The set A is i.o. tt-DNC.
(iii) The set A i.o. tt-avoids c.e. traces.

The three assertions remain equivalent if one replaces in the two last assertions truth-table reducibility by weak truth-table reducibility.

7 Computable traces and total machines

We have seen above that traceability notions defined in terms of c.e. traces can be characterized by concepts such as autocomplexity that relate to the plain Kolmogorov complexity of the initial segments of a set. We will see now that these characterizations can be extended to traceability notions defined in terms of computable traces if one considers the complexity of initial segments with respect to total machines.

Remark 26. Bienvenu and Merkle [2] have defined the notion of decidable machines, that is, machines whose domain is decidable. Obviously, every total machine is decidable, and every decidable machine can be easily converted into a total machine by first deciding whether a string is in the domain and then executing the machine as normal if that is the case, and outputting a constant otherwise.

Definition 27. *A set A is* totally complex *iff there is a computable function g such that for all total machines M and almost all n, we have $\mathrm{C}_M(A \upharpoonright g(n)) \geq n$. A set A is* totally i.o. complex *iff there is a computable function g such that for all total machines M there are infinitely many n where $\mathrm{C}_M(A \upharpoonright g(n)) \geq n$.*

Theorem 28 can be obtained from a result of Kjos-Hanssen et al. [9, Theorem 5.1] and Theorem 6. We omit the proof of Theorem 28 and give instead the very similar proof of its infinitely often version Theorem 29. In connection with the latter theorem, note that Franklin and Stephan [6] considered computably tt-traceable sets, that is, sets that do not i.o. tt-avoid computable traces, and showed that these sets are exactly the Schnorr-trivial sets.

Theorem 28. *A set A is totally complex if and only if A tt-avoids computable traces.*

Theorem 29. *A set A is totally i.o. complex if and only if A i.o. tt-avoids computable traces.*

Proof. First assume that A is not totally i.o. complex, i.e., for any computable function g there is a total machine M such that for almost all n, we have $C_M(A \upharpoonright g(n)) \leq n$. Fix any function $f \leq_{\mathrm{tt}} A$ and some tt-reduction witnessing this fact, which has use bound $u(n)$. By assumption on A, there is a total machine M such that for almost all n, we have $C_M(A \upharpoonright u(n)) \leq n$. In order to obtain a computable trace $(T_n)_n$ for f that is bounded by the function $n \mapsto 2^{n+1}$, execute all codes of length up to n on M, view the outputs as initial segments of oracles, and let T_n contain all values that one obtains by simulating the fixed tt-reduction for computing f at place n with any of these oracles. Then $f(n)$ is contained in T_n for almost all n. Since the bound 2^{n+1} on the size of the sets T_n does not depend on f, the set A is computably tt-traceable.

Next assume that A does not i.o. tt-avoid computable traces, i.e., that A is is computably tt-traceable, and recall that by the discussion preceding Remark 5 we can assume that any function wtt-reducible to A has a computable trace that is n-bounded. Given a computable function g, we need to show that there is a total machine M such that for almost all n, we have $C_M(A \upharpoonright g(n)) \leq n$. We can assume that the function $n \mapsto A \upharpoonright g(n)$ has a computable trace $(T_n)_n$ where T_n has size at most n. Let M be the machine, which on input (n, i) outputs the i-th element of T_n, if this element exists, and outputs some constant otherwise. Since the set T_n has size at most n and its canonical index can be computed from n, M is total and satisfies $C_M(A \upharpoonright g(n)) \leq 2 \log n \leq n$ for almost all n. □

8 Characterizing i.o. complex and i.o. autocomplex via lower bounds on the complexity of initial segments

When introducing the notions of i.o. complex and i.o. autocomplex, we have argued that it does not make sense to define these notions by requiring for the set A under consideration that for a computable or A-computable order,

respectively, infinitely often the order provides a lower bound for the plain Kolmogorov complexity of an initial segment of A, and the reason for this was simply that by choosing a small enough order this condition would be trivially satisfied by all sets. We will argue in this section, however, that equivalent definitions in terms of lower bounds for the complexity of initial segments can be given if plain Kolmogorov complexity C is replaced by appropriate variants, e.g., by uniform or monotonic complexity (see Li and Vitányi [10] for a more detailed account of these notions). Due to space considerations, we will restrict attention to the concept of i.o. autocomplex.

Definition 30. *Let U be the universal Turing machine used to define plain Kolmogorov complexity* C.

The length-conditioned complexity $\mathrm{C}(w \mid n)$ *of w is the length of the shortest program p such that U on input $(p, |w|)$ will output w.*

The uniform complexity $\mathrm{C}(w; n)$ *of w is the length of the shortest program p such that for all $i \leq |w|$, U on input (p, i) will output the first i bits of w, while U may do anything on inputs (p, i) with $|w| < i$.*

The monotonic complexity $\mathrm{C}_{\mathrm{mon}}(w)$ *is the length of the shortest program p such that U on input p will output some extension of w.*

From these definitions, the following chain of inequalities is immediate,

$$\mathrm{C}(w \mid n) \leq^{+} \mathrm{C}(w; n) \leq^{+} \mathrm{C}_{\mathrm{mon}}(w) \leq^{+} \mathrm{C}(w) \; . \tag{1}$$

Definition 31. *A set A is* length-conditionedly i.o. autocomplex *iff there is an A-computable order h such that for infinitely many n, we have $h(n) \leq \mathrm{C}(A \upharpoonright n \mid n)$.*

A set A is uniformly i.o. autocomplex *iff there is an A-computable order h such that for infinitely many n, we have $h(n) \leq \mathrm{C}(A \upharpoonright n; n)$.*

A set A is monotonically i.o. autocomplex *iff there is an A-computable order h such that for infinitely many n, we have $h(n) \leq \mathrm{C}_{\mathrm{mon}}(A \upharpoonright n)$.*

In connection with the following theorem, recall that the first, and hence also the second and third assertion are equivalent to A not being c.e. traceable. We omit the proof of the following theorem due to space considerations.

Theorem 32. *The following assertions are equivalent.*

(i) The set A is i.o. autocomplex.
(ii) The set A is monotonically i.o. autocomplex.
(iii) The set A is uniformly i.o. autocomplex.

These three equivalent assertions are all implied by

(iv) The set A is length-conditionedly i.o. autocomplex.

9 Time bounded traceability and complexity

In this last section, we will show that for appropriately chosen notions of complexity and traceability, the relations between these two notions can be transferred to the time-bounded setting, more precisely, to a setting of polynomial time bounds.

Definition 33. *For $t \in \mathbb{N}$, let $C^t(x) := \min\{|\sigma| : U(\sigma) = x \text{ in at most } t \text{ steps}\}$.*

Consider a coding of finite sets of natural numbers where the code of a set D consists of the concatenation of the binary expansion of elements of D in the natural order, where all the bits in the binary expansions are doubled and the binary expansions are separated from each other by the word 01. In the sequel, we will identify a finite set D with its code. Instead of looking at the Kolmogorov complexity of initial segments we will examine the Kolmogorov complexity of strings $A \upharpoonright D$ where D is a finite subset of $\mathbb{N}$. This will be defined in the straightforward way.

Definition 34. *A set A is* i.o. poly-complex *iff there is a computable order h such that for all polynomials p there are infinitely many sets D where we have for $t = p(|D| + |\max D|)$ that $\mathrm{C}^t(A \upharpoonright D \mid D) \geq h(\max D)$.*

Definition 35. *A set A is polynomial-time tt-traceable iff for all computable orders h, we have that for every function $f \leq_{\mathrm{tt}}^{\mathbf{P}} A$ there is an h-bounded trace $(T_n)_n$ such that for given n, the list of elements of T_n can be computed (or, say, printed) in time polynomial in the length of n.*

Theorem 36. *The following statements are equivalent.*

(i) A is not i.o. poly-complex.
(ii) A is polynomial-time tt-traceable.

Proof. (i) implies (ii): Let h be the desired trace bound, where we can assume $h(n) \leq n$ by switching to a delayed version of h, and let $f \leq_{\mathrm{tt}}^{\mathbf{P}} A$ be the function to be traced. Let q be the polynomial time bound of some fixed tt-reduction from f to A, and let $D(n)$ be the query set of this reduction at place n, where we can assume that $D(n)$ always contains n.

Now the mapping $g\colon n \mapsto \lfloor \log h(n) \rfloor$ is surely a computable order, so we know by assumption that for some p and almost all n we have for $t = p(|D(n)| + |\max D(n)|)$ that $\mathrm{C}^t(A \upharpoonright D(n) \mid D(n)) < g(n)$. Since t and $g(n)$ are both polynomial in the length of n, polynomial time in the length of n suffices to run the universal machine on all programs p of length strictly less than $g(n)$ with conditioning $D(n)$ for at most t steps each, interpreting the outputs obtained this way as oracles and to simulate the given reduction at place n with all of these oracles in order to obtain at most $h(n) \leq 2^{g(n)} - 1$ values that are put into the set T_n.

(ii) implies (i): We have to show for a given computable order h that there is a polynomial p such that for almost all finite sets D it holds for $t = p(|D| + |\max D|)$ that $\mathrm{C}^t(A \upharpoonright D \mid D) < h(\max D)$. Let f be the function which maps n to $A \upharpoonright D$, for all n that are a code for some finite set D, and let $f(n) = 0$ in case n is not such a code. By definition of the coding, computing $f(n)$ from A requires at most $\log n$ queries to A of length at most $\log n$. So $f \leq_{\mathrm{tt}}^{\mathbf{P}} A$, say with polynomial time bound q.

Since the length of the code for a finite set D is effectively bounded in $\max D$, we can fix a computable order h' such that for any finite set D, we have $h'(|D|) \leq$

$h(\max D)$. By assumption on A, let $(T_n)_n$ be an h'-bounded trace for f with polynomial time bound, i.e., for any finite set D with code n the value $f(n) = A \upharpoonright D$ occurs among the at most $h'(n) \leq h(\max D)$ elements of T_n and $\mathrm{C}^t(A \upharpoonright D \mid D) \leq h(\max D)$ with t polynomial in $|D| + |\max D|$, as desired. □

References

1. George Barmpalias, Rod Downey, and Noam Greenberg. K-trivial degrees and the jump-traceability hierarchy. *Proc. Amer. Math. Soc.*, 137(6):2099–2109, 2009.
2. Laurent Bienvenu and Wolfgang Merkle. Reconciling data compression and Kolmogorov complexity. In *Proceedings of the International Conference on Automata, Languages and Programming (ICALP)*, volume 4596 of *Lecture Notes in Comput. Sci.*, pages 643–654, Berlin, 2007. Springer.
3. Peter Cholak, Rod Downey, and Noam Greenberg. Strong jump-traceability I: The computably enumerable case. *Advances in Mathematics*, 217(5):2045 – 2074, 2008.
4. Rod Downey and Denis Hirschfeldt. *Algorithmic Randomness*. Springer. To appear.
5. Johanna Franklin, Noam Greenberg, Frank Stephan, and Guohua Wu. Anti-complexity, lowness and highness notions, and reducibilities with tiny use. Manuscript, 2009.
6. Johanna Franklin and Frank Stephan. Schnorr trivial sets and truth-table reducibility. *Journal of Symbolic Logic*, 75:501–521, 2010.
7. Carl G. Jockusch, Jr. Degrees of functions with no fixed points. In *Proceedings of the Eighth International Congress of Logic, Methodology and Philosophy of Science (Moscow, 1987)*, volume 126 of *Stud. Logic Found. Math.*, pages 191–201. North-Holland, Amsterdam, 1989.
8. Max I. Kanovich. On the complexity of enumeration and decision of predicates. *Soviet Math. Dokl.*, 11:17–20, 1970.
9. Bjørn Kjos-Hanssen, Wolfgang Merkle, and Frank Stephan. Kolmogorov complexity and the recursion theorem. *Trans. Amer. Math. Soc.* In print.
10. Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 2008.
11. Frank Stephan. Private communication. April 2010.
12. Sebastiaan A. Terwijn and Domenico Zambella. Computational randomness and lowness. *The Journal of Symbolic Logic*, 66(3):1199–1205, 2001.