Generic computability, and the strange results that arise in its study

Gregory Igusa

University of California, Berkeley

March 7, 2013

Outline

1. Introduction
2. Nonexistence of minimal pairs
3. Generic reducibility

Background

In complexity theory, it has been observed that problems can be difficult in theory while being quite easy to solve in practice. (Think: minesweeper)

1986: Levin introduces “average-case complexity.”

2003: Kapovich, Miasnikov, Schupp and Shpilrain introduce “generic-case complexity.”

Background

One of the advantages of generic-case complexity is that the generic-case complexity of a problem can be found without finding the worst-case complexity of the problem.

More importantly for our purposes, the generic-case complexity of a problem can sometimes be found, even if the problem is not solvable in the worst case.

For instance, for Boone's group, the word problem is strongly generically linear time. (KMSS 2003)

(Incidentally, it is unknown whether or not the word problem for finitely generated groups is always generically computable.)

Generic computablity

In recent work (∼2009), Downey, Jockusch, and Schupp introduce and analyze the notion of generic computability. A real is genericaly computable if it is possible to compute the *majority* of the bits of the real, in the following sense:

Definition

A real A is density-1 if the limit of the densities of its initial segments is 1, or in other words, if $\lim_{n\to\infty} \frac{|A\restriction n|}{n} = 1$.

Generic computablity

Definition

A real A is generically computable if there exists a partial recursive function φ whose domain is density-1 such that if $\varphi(n) = 1$ then $n \in A$, and if $\varphi(n) = 0$ then $n \notin A$.

So for example, any subset of the powers of 2 is generically computable.

As another example, a density-1 real is generically computable if and only if it has a density-1 subset which is r.e.

In fact, for A to be generically computable, it is neither necessary, nor sufficient, for A to agree with a recursive set on a set of density-1. (JS 2012)

Relative generic computability

We now relativize the notion of generic computability:

Definition

For reals A and B, A is generically B-computable if A is generically computable using B as an oracle. In this case, we write $B \to_g A$.

We do not alter the objects that are being computed, nor do we alter the procedures that we are allowed to use. Rather, we alter what it means to compute an object.

As a result, this notion of relative computability is highly non-transitive.

In fact, one can fairly easily show that any countable reflexive binary relation embeds into the reals under $\to_g$:

Embedding relations into generic computation

Proposition

For any reflexive binary relation R on the natural numbers $\mathbb{N}$, there exists a subset S of $\mathbb{R}$ such that $\langle \mathbb{N}, R \rangle$ is isomorphic to $\langle S, \rightarrow_g \rangle$.

The ideas behind the proof are fairly simple:

Every one of the reals in S is broken into a "large" part and a "small" part.

On the "large" part, we code what is necessary to generically compute the real, and on the "small" part, we code the join of all the reals that the real is supposed to be able to generically compute.

The correct way to think of this is not that $\rightarrow_g$ is a messy relationship, but rather that the inputs and outputs of a generic computation are not really the same kind of object.

No degree structure: the reals that A can generically compute have no reason to depend on the reals that can generically compute A.

"$A \rightarrow_g B$" is a statement that depends entirely on the Turing degree of A.
(And on the generic degree of B, defined later.)

Minimal pairs

Thus, we mostly concern ourselves with questions in which a given real is never treated as both an input and an output of a generic computation.

Here, we consider the question of existence of minimal pairs.

Theorem (I.)

There do not exist minimal pairs for generic computation.

In fact, the proof can be strengthened to show:

Theorem (I.)

For any n, and any nonrecursive reals $A_0, \ldots, A_{n-1}$, there exists a real C such that C is not generically computable, but such that for every i, $A_i \rightarrow_g C$

The following proposition was a key step in the proof of the nonexistence of minimal pairs.

Proposition (I.)

For any nonrecursive real A, there is a density-1 set that is r.e. in A, and that has no density-1 r.e. subset.

Strictly speaking, this proposition is a corollary to the theorem, since a counterexample to this proposition would necessarily be half of a minimal pair, but the proof generalizes to prove the nonexistence of minimal pairs, and indeed, the proof is basically the case $n = 1$ of the proof of the "minimal sets" theorem.

We briefly sketch the techniques used in this proof.

Proof (sketch).

We construct φ so that if A is nonrecursive, then φ^A computes a density-1 set with no density-1 r.e. subset.
Our primary technique will be setting "traps" of the following form.

- Choose some $\sigma \in 2^\omega$.
- Create a large gap in φ^X for every $X \succ \sigma$.
- Wait.

At this point, an r.e. set W has two choices:

Either it enumerates some of the elements of our gap, ensuring that it can never be a subset of φ^X for any $X \succ \sigma$.

Or it avoids the gap, causing its density to drop to one more time, while we only injured the X's extending σ. □

Proof (sketch, continued).

The rest of the construction is simply ensuring that if X has infinitely many traps put on it, then X is computable.

Hence, somewhat counterintuitively, we do not use the nonrecursiveness of A to ensure that, φ^A has no density-1 r.e. subset, but rather simply to ensure that φ^A is density-1.

(Bearing in mind that these computations are secretly generic computations of density-1 sets, we use nonrecursiveness of A not to ensure that φ^A is not "computable," but rather to ensure that φ^A is "total.") □

Generic reducibility

We now discuss generic reducibility, a notion of generic computation that has been modified to make it transitive.

We want the definition of generic reducibility to satisfy the following two requirements:

- If $A \leq_g B$ and $B \leq_g C$ then $A \leq_g C$
- A is generically computable if and only if $A \leq_g 0$

As such, we make the following definition:

Generic reducibility

Definition

We say A generically reduces to B if from *any* density-1 subset of the bits of B, one can generically compute A. In this case, we write $A \leq_g B$.

The definition is intentionally left vague, as there are multiple provably distinct ways to formalize this, with the most important distinction being whether or not the reduction must be uniform over all generic oracles for B.

Unless otherwise specified, everything said in this talk holds for all of the different formalizations.

Embedding the Turing degrees in the generic degrees

It turns out that one can embed the Turing degrees in the generic degrees:

Definition

For any real X, let $\mathcal{R}(X)$ be defined by: $n \in \mathcal{R}(X) \leftrightarrow m \in X$, where 2^m is the largest power of 2 dividing n.

So we have "stretched" every bit of X into an entire "column" of $\mathcal{R}(X)$.

Since every generic description of $\mathcal{R}(X)$ must include at least one bit from every column, it must be able to compute X.

As a result, generically computing $\mathcal{R}(X)$ is the same as computing X, and working with $\mathcal{R}(X)$ as a generic oracle is the same as working with X as an oracle in the usual sense.

Π^1_1-completeness

One of the biggest obstacles to working with generic reduction is simply the complexity of the definition.

Theorem (I.)

Uniform generic reduction is Π^1_1-complete.

So there is no “easy” way around the universal quantifier in the definition.

The proof uses a recursion theoretic analogue of an irreversible function.

Degrees of density-1 sets

One somewhat fruitful line of inquiry has been the investigation of the generic degrees of density-1 sets. Call these degrees density-1 degrees.

Frequently, the strange things that happen when studying generic computation come up entirely through the study of the halting sets of the computations, and this seems like a natural way to distill out and study that aspect of the theory.

By the proof of our theorem about minimal pairs, we know that density-1 degrees can be found below the embedded image of any Turing degree.

We can also make density-1 degrees that compute slow growing functions, so we can make density-1 degrees that are above anything that can be computed from a sufficiently slow growing function.

We can also prove some very comforting results about these degrees: for example, the density-1 degrees are dense!

Proposition

Given any two density-1 degrees $\mathbf{a}$ *and* $\mathbf{b}$*, if* $\mathbf{a} >_g \mathbf{b}$*, then there exists a density-1 degree* $\mathbf{c}$*, such that* $\mathbf{a} >_g \mathbf{c} >_g \mathbf{b}$*.*

We can even find a pair of intermediate degrees whose join is the larger degree.

Minimal degrees and pairs

Now, we turn our attention to two open questions.

Question

Are there any minimal generic degrees?

Question

Are there minimal pairs in the generic degrees?

We also ask one additional question:

Question

Given a nonzero generic degree $\mathbf{a}$*, is there always a density-1 degree* $\mathbf{b}$ *such that is* $\mathbf{a} \geq_g \mathbf{b}$*?*

If the answer to the question is “yes,” then there cannot be any minimal generic degrees, because the density-1 degrees are dense.

On the other hand, this question is the analogue of the key proposition from the minimal pairs argument for generic computation, so if the answer is “no,” then the counterexample is half of a minimal pair for generic reduction.

From these two observations, we get a free result:

Corollary

If there are minimal generic degrees, then there are minimal pairs of generic degrees.

End

Thank you