Nonexistence of Minimal Pairs for Generic Computability

Gregory Igusa

University of California, Berkeley

July 2, 2011

Outline of talk

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

Generic computablity

In recent work, 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 $\lim_{n\to\infty} \frac{|A\upharpoonright n|}{n} = 1$.

Generic computablity

In recent work, 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.

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.

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.

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.

Unlike with most reductions, we alter the *output*, not the *procedure*.

Thus, this notion of relative computability is highly non-transitive.

In fact, one can fairly easily show that any countable reflexive binary relation can be realized as a subset of the reals under relative generic computability.

Later, we will also discuss *generic reducibility*, a related transitive notion.

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.

Unlike with most reductions, we alter the *output*, not the *procedure*.

Thus, this notion of relative computability is highly non-transitive.

In fact, one can fairly easily show that any countable reflexive binary relation can be realized as a subset of the reals under relative generic computability.

Later, we will also discuss *generic reducibility*, a related transitive notion.

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.

Unlike with most reductions, we alter the *output*, not the *procedure*.

Thus, this notion of relative computability is highly non-transitive.

In fact, one can fairly easily show that any countable reflexive binary relation can be realized as a subset of the reals under relative generic computability.

Later, we will also discuss *generic reducibility*, a related transitive notion.

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.

Unlike with most reductions, we alter the *output*, not the *procedure*.
Thus, this notion of relative computability is highly non-transitive.
In fact, one can fairly easily show that any countable reflexive binary relation can be realized as a subset of the reals under relative generic computability.
Later, we will also discuss *generic reducibility*, a related transitive notion.

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.

Unlike with most reductions, we alter the *output*, not the *procedure*.
Thus, this notion of relative computability is highly non-transitive.
In fact, one can fairly easily show that any countable reflexive binary relation can be realized as a subset of the reals under relative generic computability.
Later, we will also discuss *generic reducibility*, a related transitive notion.

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.

Unlike with most reductions, we alter the *output*, not the *procedure*.
Thus, this notion of relative computability is highly non-transitive.
In fact, one can fairly easily show that any countable reflexive binary relation can be realized as a subset of the reals under relative generic computability.
Later, we will also discuss *generic reducibility*, a related transitive notion.

Minimal pairs

Non-transitivity of relative generic computation seems like it would make most recursion theoretic questions meaningless, but many questions don't rely on transitivity to be formed.

Here, we concern ourselves with the existence (or non-existence) of minimal pairs.

Theorem (L)

For any nonrecursive reals A and B there exists a real G such that G is not generically computable, but such that G is both generically A-computable and generically B-computable.

Note that the case where A and B are both Δ^0_2 is proved by Downey, Jockusch, and Schupp.

Minimal pairs

Non-transitivity of relative generic computation seems like it would make most recursion theoretic questions meaningless, but many questions don't rely on transitivity to be formed.

Here, we concern ourselves with the existence (or non-existence) of minimal pairs.

Theorem (I.)

For any nonrecursive reals A and B there exists a real C such that C is not generically computable, but such that C is both generically A-computable and generically B-computable.

Note that the case where A and B are both Δ^0_2 is proved by Downey, Jockusch, and Schupp.

Minimal pairs

Non-transitivity of relative generic computation seems like it would make most recursion theoretic questions meaningless, but many questions don't rely on transitivity to be formed.

Here, we concern ourselves with the existence (or non-existence) of minimal pairs.

Theorem (I.)

For any nonrecursive reals A and B there exists a real C such that C is not generically computable, but such that C is both generically A-computable and generically B-computable.

Note that the case where A and B are both Δ^0_2 is proved by Downey, Jockusch, and Schupp.

Minimal pairs

Non-transitivity of relative generic computation seems like it would make most recursion theoretic questions meaningless, but many questions don't rely on transitivity to be formed.

Here, we concern ourselves with the existence (or non-existence) of minimal pairs.

Theorem (I.)

For any nonrecursive reals A and B there exists a real C such that C is not generically computable, but such that C is both generically A-computable and generically B-computable.

Note that the case where A and B are both Δ^0_2 is proved by Downey, Jockusch, and Schupp.

Proof sketch

To prove this, we show the following:

Proposition

For any nonrecursive reals A and B there exist Turing functionals φ and ψ such that φ^A and ψ^B are both density-1 sets, but such that the union $\varphi^A \cup \psi^B$ contains no density-1 r.e. subset.

Proof sketch (cont.)

- We construct Turing functionals φ and ψ such that for *any* reals X and Y, the union $\varphi^X \cup \psi^Y$ contains no density-1 r.e. subset.
- In doing so, there will be countably many reals X_i and Y_i such that φ^{X_i} and ψ^{Y_i} are not density-1.
- The specifics of the construction ensure that all of these X_i and Y_i are Δ^0_2.
- A slight modification of the construction works if one or both of A and B is Δ^0_2.

Proof sketch (cont.)

- We construct Turing functionals φ and ψ such that for *any* reals X and Y, the union $\varphi^X \cup \psi^Y$ contains no density-1 r.e. subset.
- In doing so, there will be countably many reals X_i and Y_i such that φ^{X_i} and ψ^{Y_i} are not density-1.
- The specifics of the construction ensure that all of these X_i and Y_i are Δ^0_2.
- A slight modification of the construction works if one or both of A and B is Δ^0_2.

Proof sketch (cont.)

- We construct Turing functionals φ and ψ such that for *any* reals X and Y, the union $\varphi^X \cup \psi^Y$ contains no density-1 r.e. subset.
- In doing so, there will be countably many reals X_i and Y_i such that φ^{X_i} and ψ^{Y_i} are not density-1.
- The specifics of the construction ensure that all of these X_i and Y_i are Δ^0_2.
- A slight modification of the construction works if one or both of A and B is Δ^0_2.

Proof sketch (cont.)

- We construct Turing functionals φ and ψ such that for *any* reals X and Y, the union $\varphi^X \cup \psi^Y$ contains no density-1 r.e. subset.
- In doing so, there will be countably many reals X_i and Y_i such that φ^{X_i} and ψ^{Y_i} are not density-1.
- The specifics of the construction ensure that all of these X_i and Y_i are Δ^0_2.
- A slight modification of the construction works if one or both of A and B is Δ^0_2.

Generic reducibility

We now discuss generic reducibility, the implications of our result in terms of generic reducibility, and some questions on the subject.

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

We now discuss generic reducibility, the implications of our result in terms of generic reducibility, and some questions on the subject.

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

We now discuss generic reducibility, the implications of our result in terms of generic reducibility, and some questions on the subject.

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

We now discuss generic reducibility, the implications of our result in terms of generic reducibility, and some questions on the subject.

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

We now discuss generic reducibility, the implications of our result in terms of generic reducibility, and some questions on the subject.

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, and everything that will be said for the rest of this talk holds equally well for any such formalization.

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, and everything that will be said for the rest of this talk holds equally well for any such formalization.

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.

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.

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.

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.

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.

Quasi-minimal generic degrees

As a corollary of our result about minimal pairs, we can conclude that below any nonzero generic degree, there is a degree that is not above the image of any nonzero Turing degree, under the embedding induced by $X \mapsto \mathcal{R}(X)$.

We first define quasi-minimal generic degrees:

Definition

A generic degree a is quasi-minimal if a is nonzero, and if for every nonrecursive real X, the generic degree of $\mathcal{R}(X^{\mathcal{O}})$ is not below a.

Quasi-minimal generic degrees

As a corollary of our result about minimal pairs, we can conclude that below any nonzero generic degree, there is a degree that is not above the image of any nonzero Turing degree, under the embedding induced by $X \mapsto \mathcal{R}(X)$.

We first define quasi-minimal generic degrees:

Definition

A generic degree $\mathbf{a}$ is quasi-minimal if $\mathbf{a}$ is nonzero, and if for every nonrecursive real X, the generic degree of $\mathcal{R}(X)$ is not below $\mathbf{a}$.

Quasi-minimal generic degrees

As a corollary of our result about minimal pairs, we can conclude that below any nonzero generic degree, there is a degree that is not above the image of any nonzero Turing degree, under the embedding induced by $X \mapsto \mathcal{R}(X)$.

We first define quasi-minimal generic degrees:

Definition

A generic degree **a** is quasi-minimal if **a** is nonzero, and if for every nonrecursive real X, the generic degree of $\mathcal{R}(X)$ is not below **a**.

Quasi-minimal generic degrees

Using this notation, we rephrase the corollary:

Proposition

For every nonzero generic degree **a**, *there is a quasi-minimal generic degree* **b** *such that* $\mathbf{a} \geq_G \mathbf{b}$.

Proof.

If **a** is not quasi-minimal, choose X such that the generic degree of $\mathcal{R}(X)$ is below **a**. Choose Y so that X and Y form a minimal pair for Turing reduction. Since X and Y do not form a minimal pair for generic computability, there is a generic degree **b** that is below both $\mathcal{R}(X)$ and $\mathcal{R}(X)$. This degree must be quasi-minimal. □

Quasi-minimal generic degrees

Using this notation, we rephrase the corollary:

Proposition

For every nonzero generic degree **a**, *there is a quasi-minimal generic degree* **b** *such that* $\mathbf{a} \geq_G \mathbf{b}$.

Proof.

If **a** is not quasi-minimal, choose X such that the generic degree of $\mathcal{R}(X)$ is below **a**. Choose Y so that X and Y form a minimal pair for Turing reduction. Since X and Y do not form a minimal pair for generic computability, there is a generic degree **b** that is below both $\mathcal{R}(X)$ and $\mathcal{R}(X)$. This degree must be quasi-minimal. □

Questions

Naturally, the main question to ask is whether there is a minimal pair for generic reduction. By the previously mentioned corollary, if there is such a minimal pair, there would have to be one in which both of the degrees were quasi-minimal.

Looking from the opposite direction, it would also be interesting to know whether the join of two quasi-minimal generic degrees must be quasi-minimal.

Probably before any of this can be addressed, we would require more machinery for constructing quasi-minimal degrees, or a better characterization of them.

Questions

Naturally, the main question to ask is whether there is a minimal pair for generic reduction. By the previously mentioned corollary, if there is such a minimal pair, there would have to be one in which both of the degrees were quasi-minimal.

Looking from the opposite direction, it would also be interesting to know whether the join of two quasi-minimal generic degrees must be quasi-minimal.

Probably before any of this can be addressed, we would require more machinery for constructing quasi-minimal degrees, or a better characterization of them.

Questions

Naturally, the main question to ask is whether there is a minimal pair for generic reduction. By the previously mentioned corollary, if there is such a minimal pair, there would have to be one in which both of the degrees were quasi-minimal.

Looking from the opposite direction, it would also be interesting to know whether the join of two quasi-minimal generic degrees must be quasi-minimal.

Probably before any of this can be addressed, we would require more machinery for constructing quasi-minimal degrees, or a better characterization of them.

End

Thank you!