Natural Definability in the Nether Regions of the Computably Enumerable Degrees

Rod Downey Noam Greenberg

May 24, 2006

Computable sets and functions

Definitions

An algorithm is a program written in Basic.

A function f is computable if there is some algorithm which, given an input x, outputs $f(x)$.

A set A is computable if there is some algorithm which, given an input x, answers the question "is x in A?"

Computable sets and functions

Definitions

An algorithm is a program written in Pascal.

A function f is computable if there is some algorithm which, given an input x, outputs $f(x)$.

A set A is computable if there is some algorithm which, given an input x, answers the question "is x in A?"

Computable sets and functions

Definitions

An algorithm is a program written in C.

A function f is computable if there is some algorithm which, given an input x, outputs $f(x)$.

A set A is computable if there is some algorithm which, given an input x, answers the question "is x in A?"

Computable sets and functions

Definitions

An algorithm is a program written in Python.

A function f is computable if there is some algorithm which, given an input x, outputs $f(x)$.

A set A is computable if there is some algorithm which, given an input x, answers the question "is x in A?"

Computable sets and functions

Definitions

An algorithm is a program written in your favourite language.

A function f is computable if there is some algorithm which, given an input x, outputs $f(x)$.

A set A is computable if there is some algorithm which, given an input x, answers the question "is x in A?"

Examples

- The set of even numbers.
- The set of prime numbers.
- The function which, given a number, returns its prime divisors.
- The collection of graphs that have Hamiltonian cycles.
- The function which, given a polynomial p (with rational coefficients), halts if p has an integral solution, in which case it returns one such solution.

Computably enumerable sets

Definition

A set is computably enumerable if there is some algorithm that outputs its elements.

Theorem

The following are equivalent for an infinite set A:

1. *A is computably enumerable;*
2. *A is the range of some computable function;*
3. *A is the domain of some computable function.*

Examples

- Any computable set.
- The collection of polynomials that have integral solutions.
- The *word set*: the collection of pairs $(\bar{g}, \bar{r})$ of generators and relations such that the generated group is trivial.
- The *halting set*: the collection of algorithms which terminate.

In fact:

Theorem

A set is computable if, and only if, both it and its complement are computably enumerable.

THE "UNIVERSAL MACHINE"

There is an algorithm `run` which takes as input a pair (M, x) and activates the algorithm M on input x; and if it halts, returns the output of that computation. The universal machine is the function computed by this algorithm.

THE "UNIVERSAL MACHINE"

There is an algorithm `run` which takes as input a pair (M, x) and activates the algorithm M on input x; and if it halts, returns the output of that computation. The universal machine is the function computed by this algorithm.

EXERCISE

Find an algorithm that outputs itself.

THEOREM

The domain of the universal machine is not computable.

PROOF.

Suppose it is. Devise the following algorithm M. As input, it takes an algorithm N that takes one input. If N does not halt when run on itself (as input), then M halts; but if N does halt when run on itself, then M enters an infinite loop.

Now ask: does M halt when run on itself? □

The domain of the universal machine is denoted by $\emptyset'$.

Relative computability

Definitions

Let A be a set. An algorithm with oracle A is an algorithm in your favourite language, enhanced by a procedure that decides membership in A.

A set B is computable from A if there is an algorithm with oracle A that decides membership in B. We write $B \leqslant_T A$.

Relative computability is a transitive relation.

Examples

- A computable set B is computable from any oracle.
- Any computably enumerable set is computable from the domain of the universal machine.
- If a set B is computable from a computable set, then B itself is computable.

TURING DEGREES

DEFINITIONS

Two sets A and B are Turing equivalent if A is computable from B and B is computable from A.

The Turing degree of a set A is the collection of all sets B that are Turing equivalent to A.

Turing reducibility induces a partial ordering on the collection of Turing degrees.

COMPUTABLY ENUMERABLE DEGREES

The collection of the computable sets forms the least Turing degree, denoted by **0**. We already know that $\emptyset'$ is not computable; thus its degree, $\mathbf{0}'$, is strictly greater than **0**. Also, for any degree **a** that contains a c.e. set, $\mathbf{a} \leqslant \mathbf{0}'$.

Are there any c.e. degrees apart from **0** and $\mathbf{0}'$? Various c.e. sets have been shown to be non-computable; prominent examples are the word set and the collection of polynomials with integral solutions (Hilbert's 10^{th} problem). However, they all have degree $\mathbf{0}'$.

In 1956, Friedberg and Muchnik have independently shown that there are incomparable c.e. degrees. For this, they introduced the powerful *priority method*. Using this method, the c.e. degrees have been extensively investigated. For example, it is known (Sacks) that the c.e. degrees are dense.

Open question

Is there an automorphism of the c.e. degrees?

We denote the structure of the c.e. degrees by $\mathcal{R}$.

Definable classes

Definition

A set $C \subset \mathcal{R}$ is definable if it is invariant under the automorphisms of $\mathcal{R}$.

This concept may be too broad... we restrict ourselves to "nicely" definable classes.

Example

The collection of degrees $\mathbf{a} \in \mathcal{R}$ such that for some $\mathbf{b} \in \mathcal{R}$ we have $\mathbf{a} \vee \mathbf{b} = \mathbf{0}'$.

Arithmetical definability

A collection of c.e. sets is arithmetically definable if they share some property.

Examples

- The collection of infinite c.e. sets.
- Simple sets: c.e. sets whose complement does not contain an infinite c.e. sets.
- Promptly simple sets: simple sets for which witnesses for simplicity are obtained “quickly”.

A deeper understanding of the degrees (and of the notion of computation itself) is obtained by making connections between an arithmetically definable collection of c.e. sets and the collection of degrees that contain them.

EXAMPLE (AMBOS-SPIES, JOCKUSCH, SHORE, SOARE)

A c.e. degree **a** contains a promptly simple set iff there is some c.e. degree **b** such that $\mathbf{a} \wedge \mathbf{b} = \mathbf{0}$.

The Turing jump operator

Given a set A, the enhanced programming language that uses A as an oracle has a universal machine of its own. Its domain is denoted by A'.

Theorem

If $A \leqslant_T B$ then $A' \leqslant_T B'$.

Thus, if sets A and B are Turing equivalent, then so are A' and B'. The map $A \mapsto A'$ induces an order-preserving function from the degrees to the degrees.

Theorem

For any degree $\mathbf{a}$, $\mathbf{a} < \mathbf{a}'$.

JUMP CLASSES

Let **a** and **b** be degrees. We let $\mathbf{a} \leqslant_1 \mathbf{b}$ if $\mathbf{a}' \leqslant \mathbf{b}'$. This is a transitive relation which is coarser than the usual ordering on the degrees. This can be iterated: $\mathbf{a} \leqslant_2 \mathbf{b}$ iff $\mathbf{a}' \leqslant_1 \mathbf{b}'$ iff $\mathbf{a}'' \leqslant \mathbf{b}''$, etc.

$\leqslant_n$- equivalence classes are called jump classes. In the c.e. degrees, the ones that are used mostly are the least and greatest jump classes: **a** is low_n if $\mathbf{a} \equiv_n \mathbf{0}$; it is high_n if $\mathbf{a} \equiv_n \mathbf{0}'$.

THE NETHER REGIONS OF THE C.E. DEGREES

Low and low_2 degrees were previously thought to be simple and boring: they are too close to being computable.

Recent research shows a plethora of fascinating phenomena and subclasses in the lower regions of the c.e. degrees.

EXAMPLES

- Array computable degrees (Downey, Jockusch, Stob): combinatorial properties of c.e. sets.
- Almost deep degrees (Cholak, Groszek, Slaman): low degrees $\mathbf{a}$ such that for any other low degree $\mathbf{b}$, $\mathbf{a} \vee \mathbf{b}$ is low.
- K-trivial degrees (Downey, Hirschfeldt, Nies): connections with measure theory.

A UNIFORM DEFINABILITY RESULT

Nies, Shore and Slaman obtained a striking definability result by coding arithmetic in the c.e. degrees. In our terminology, they obtained:

- A definable way of associating, for each string x, a degree $\mathbf{b}_x$ such that executing algorithms is definable: for each M, the map $\mathbf{b}_x \mapsto \mathbf{b}_{M(x)}$ is elementarily definable.
- And, a definable map g from the c.e. degrees to coded algorithms, such that for any degree $\mathbf{a}$, $g(\mathbf{a}) = \mathbf{b}_M$, where M is an algorithm that enumerates a c.e. set which is $\equiv_2$ to $\mathbf{a}$.

The result on uniform definability has several corollaries:

Corollaries

- Let $C \subset \mathcal{R}$ be invariant under $\equiv_2$. Then C is definable iff it is arithmetically definable.
- Every jump class, except perhaps for low, is definable.
- The c.e. degrees are as complicated as possible.

The bi-interpretability conjecture

The above result can be improved to obtained Turing equivalence rather than just second jump class equivalence.

Corollaries of the bi-interpretability conjecture:

- Any set of degrees is definable iff it is arithmetically definable.
- The c.e. degrees are rigid, and every degree is definable.

The uniform definability result has two shortcomings:

1. The definitions are elementary but not natural.
2. Not useful for proper subclasses of the low_2 degrees.

WORKING BELOW A C.E. DEGREE

Thesis: the position of a degree in the jump-class hierarchy is reflected in its computational power.

The main problem of working below a given c.e. degree is that the construction needs to be *computable*, so appealing to an oracle is not allowed.

PERMITTING

To make a constructed c.e. set A computable from a given c.e. set B, we often use permitting: allow to enumerate x into A only if at the given stage of the construction, x or a smaller number enters B.

The "speed of the enumeration" of B influences how well we can use this technique below B.

For example:

THEOREM

Every promptly simple set bounds a minimal pair.

The flip side of speed of enumeration is the rate of growth of functions.

Namely:

A complicated set B is characterised by the property that small numbers are enumerated late into B. The settling-down function therefore grows quickly.

Domination

Definition

Let $f, g : \mathbb{N} \to \mathbb{N}$. We say that f dominates g if for all but finitely many n we have $g(n) < f(n)$.

This notion yields a notion of relative complexity: given two degrees **a** and **b**, we say that **b** is "much greater" than **a** (and write $\mathbf{a} \ll \mathbf{b}$) if there is some function computable from **b** which dominates all functions computable from **a**.
Here is a key fact, due to Martin.

Theorem

A degree **b** *is high iff* $\mathbf{0} \ll \mathbf{b}$.

Corollary

A degree **a** *is* low_2 *iff* $\mathbf{a} \ll \mathbf{0}'$.

These results, together with permitting, yield structural corollaries. For example:

Examples

- Every high degree bounds a minimal pair.
- (Downey, Shore) Every non-low$_2$ degree bounds a copy of the 1-3-1 lattice.

The problem with these results is that the converses do not hold. For example, there are low degrees that bound minimal pairs and even the 1-3-1 lattice.

We need to measure complexity in a new way.

Computable approximations

Theorem

The following are equivalent for any function f:

1. $f \leqslant_T \mathbf{0}'$.
2. *f has a computable approximation: there is some computable function h such that for all x,*

$$f(x) = \lim_{s \to \infty} h(x, s).$$

Given the “limit lemma”, a new measure of complexity can be introduced, this time asking if we can get a nice approximation. For example:

DEFINITION

A function f is ω-c.e. if it has some computable approximation $h(x,s)$ such that the “mind change” function, namely

$$l(x) = \#\{s : h(x,s+1) \neq h(x,s)\}$$

is dominated by some computable function.

TOTALLY ω-C.E. DEGREES

The notion of ω-c.e.-ness does not conform to Turing reducibility: there are ω-c.e. functions of degree $\mathbf{0}'$.

So we use brute force to reconcile between them:

DEFINITION

A c.e. degree $\mathbf{a}$ is totally ω-c.e. if every $f \leqslant_T \mathbf{a}$ is ω-c.e.

All totally ω-c.e. degrees are low_2. On the other hand, every array-computable degree is totally ω-c.e.
There are low degrees which are totally ω-c.e. and there are low degrees which are not.

A DEFINABILITY RESULT

THEOREM (DGW)

The totally ω-c.e. degrees are (naturally) definable.

The definition uses lattice embeddings. In one direction, the existence of some function f, computable from a given degree **a**, which is not ω-c.e., is used to get "enough permitting" to build an embedding below **a**.

In the other direction, we show that **a** "runs out of gas" and cannot cause disturbances (to our showing that a certain meet cannot exist), because the disturbances are computable from **a**, and therefore ω-c.e.

THEOREM

There are maximal totally ω-c.e. degrees.

This gives a definable anti-chain (incomparable set) of degrees.

TOTALLY $< \omega^\omega$-C.E. DEGREES

Ershov defined a transfinite hierarchy of complexity according to how nice computable approximations are. For our purposes, we can define inductively:

DEFINITION

A function is ω^{n+1}-c.e. if it has some computable approximation such that the mind-change function is dominated by some function which is ω^n-c.e.

This gives rise to a hierarchy in the low_2-degrees, namely that of the totally ω^n-c.e. degrees. Also,

DEFINITION

A c.e. degree **a** is totally $< \omega^\omega$-c.e. if every function computable from **a** is ω^n-c.e. for some n.

THEOREM

The totally $< \omega^\omega$-c.e. degrees are naturally definable. Indeed, a c.e. degree bounds a copy of the 1-3-1 iff it is not totally $< \omega^\omega$-c.e.

Further results

- A theorem on classical and higher computability theory.
- Further constructions are captured by the new classes.

Question

Are the totally ω^n-c.e. degrees definable?