Research Statement

Antonio Montalbán

November 14, 2010

I am interested in studying the complexity of mathematical practice. In mathematics, as we all know, some structures are more complicated than others, some constructions more complicated than others, and some proofs more complicated than others. I am interested in understanding how to measure this complexity and in measuring it. The motivations for this come from different areas. Form a foundational viewpoint, we want to know what assumptions we really need to do mathematics (ZFC set theory is way much more than we usually use), and we are also interested in knowing what assumptions are used in the different areas of mathematics. Form a computational viewpoint, it is important to know what part of mathematics can be done by mechanical algorithms, and, even for the part that can't be done mechanically, we want to know how constructive are the objects we deal with. Furthermore, it is sometimes the case that this computational analysis allows us to find connections between constructions in different areas of mathematics, and in many cases to obtain a deeper understanding of mathematical objects being analyzed.

My work is quite diverse in terms of the techniques I have used, the approaches I have taken, and the areas of mathematics that I have analyzed. However, my background area is Computability Theory, and most of my work can be considered as part of this branch of Mathematical Logic.

Inside computability theory, I have worked in various different areas. I have been particularly interested in the programs of Computable Mathematics, Reverse Mathematics and Turing Degree Theory. The former studies the computability aspects of mathematical theorems and structures. The second one analyzes the complexity of mathematical theorems in terms of the complexity of the constructions needed for their proofs. The latter studies the partial ordering induced by the relation "computable from" in an abstract way. I have also written papers in other areas like effective randomness, automata theory, the lattice of Π^0_1-classes, etc..

In this short research statement I will restrict to comment on the programs and problems I have dedicated most of my time. On my web page I have a link to my "research statement for logicians," where I explain a larger part of my work. Here, I will start by briefly describing the basic ideas behind the programs I am interested in. Then I will describe a few of my results, and concentrate only on the following four.

Theorem 1 (Montalbán, Shore 2009)**.** *Given a fixed number n, second-order arithmetic can prove that every Boolean combination of n $\mathbf{G}_{\delta\sigma}$-sets is determined. However, for each n a different proof is needed and no single proof works for all n: second-order arithmetic cannot show that all Boolean combinations of any number of $\mathbf{G}_{\delta\sigma}$-sets are determined.*

Theorem 2 (Montalbán 2005)**.** *Every hyperarithmetic linear ordering is equimorphic with a computable one. (Two linear orderings are* equimorphic *if they can be embedded in each other.)*

Theorem 3 (Harris, Montalbán 2007)**.** *For every n, there is a* finite *complete set of computably infinitary Π_n formulas for the class of Boolean algebras.*

Theorem 4 (Montalbán 2003)**.** *Every countable jump upper semilattice can be embedded into the Turing Degrees $(\mathbf{D}, \leq_T, \vee, ')$ (of course, preserving join and jump).*

Basic concepts

The main concept in computability theory is the relation "computable from". A set $A \subseteq \mathbb{N}$ is said to be *computable from* a set $B \subseteq \mathbb{N}$, and we write $A \leq_T B$, if there is a computable procedure that can tell whether an element is in A or not using B as an *oracle*, that is, we let the procedure use the information of which elements are in B. This is a very robust and natural notion that captures precisely the notion of *algorithm*. A set A is said to be *computable* if it is computable without the use of any oracle. We chose to work with subsets of $\mathbb{N}$ because it is enough: every finite object can be encoded by a single number (using, for instance, the binary representation of the number, as modern computers do). For example, strings, graphs, trees, simplicial complexes, group presentations, etc., if they are finite, they can be coded by a single number, and the method for coding is usually obvious and inessential. Here is an example: each

triangulation of a compact manifold can be encoded by a single natural number; It can then be shown that the set of numbers coding simply connected compact manifolds is not computable, implying that there is no algorithm to decide simply connectedness. But, on the other hand, this set is computable from the set of natural numbers which encode group presentations given by (generators, relations) representing a non-trivial group (this set is called the *word problem*).

Computable Mathematics. Effective mathematics is concerned with the computable aspects of mathematical objects and constructions. I have been interested on general questions like the following: When can a mathematical structure be represented computably? How difficult is it to recognize a certain structure? Can information be encoded into the isomorphism type of a structure? We search for answers that connect computational properties with structural or algebraic properties.

My research in this area has concentrated on linear orderings, well-quasi-orderings and Boolean algebras, but I have also worked with other kinds of structures like torsion-free abelian groups, vector spaces, and I recently had a student working on Artinian Rings. Another important part of my work in this area has been on finding general behaviors of interaction between structural and computational properties that work for any kind of structure. Lately I have written a few paper analyzing the relations on a structure that can be defined from a certain number of Turing jumps. In that paper I prove that there are two kinds of classes of structures: ones where we can nicely characterize all the relations defined within n Turing jumps and where no information can be encoded into the n-th jump; and ones where there is no such nice characterization but any information can be encoded into the n-th jump of some structure of the class. I will describe other, more concrete, aspects of my work in computable mathematics below.

Reverse Mathematics. Logicians have developed many ways of giving precise meaning to the notion of a theorem being more difficult or complex than another. Remarkably, in the context of reverse mathematics, it appears that a majority of the constructions we do in mathematics belong to one out of five different complexity levels. Why there are so few levels, and why there are practically no constructions more difficult than the fifth level remain unclear. Understanding why this happens is one of the guiding question for my research.

The idea behind Reverse Mathematics goes as follows. We start by fixing a basic system of axioms as a base. The most commonly used base system is called RCA_0 that essentially says that computable sets exists–this is all we assume. Now, given a theorem of "ordinary" mathematics, the question we ask is what axioms do we need to add to the basic system to prove this theorem. It is often the case in Reverse Mathematics that we can show that certain axioms are necessary to prove a theorem by showing that the axioms follow from the theorem using the basic system. Because of this idea, this program is called Reverse Mathematics. Many different systems of axioms have been defined and studied. But, as mentioned above, a very interesting fact is that most of the theorems, whose proof-theoretic strength has been analyzed, have been proved equivalent over RCA_0 to one of five systems, which we will call the *main five systems*.

The programs of Reverse Mathematics and Computable Mathematics are closely related. The reason is the following: Many of the main axiom systems of second-order arithmetic are equivalent to statements of the form "sets of a certain computational complexity exist", and also to statements of the form "constructions of a certain type are allowed". So, when we study the proof-theoretic strength of a theorem, many times we end up studying the complexity of the constructions in the proof of that theorem, and the complexity of the objects involved in the proof. As we said above, this is also what we do in Computable Mathematics.

I have recently written a paper where I describe what I think are the main open questions in Reverse Mathematics.

Turing Degree Theory. The structure of the Turing degrees is defined as follows. The relation $\leq_T$ (defined above) is a quasi-ordering on $\mathcal{P}(\mathbb{N})$, the set of subsets of $\mathbb{N}$. It induces an equivalence relation ($A \equiv_T B \iff A \leq_T B \ \& \ B \leq_T A$) and a partial ordering on the equivalence classes. The equivalence classes are called *Turing degrees*, and we let $\mathbf{D}$ be the set of all the Truing degrees. With the intention of studying the relation $\leq_T$ abstractly, one of the main goals of Computability Theory is to understand the shape of the structure of $(\mathbf{D}, \leq_T)$.

Various approaches have been taken to understanding the shape of the Turing Degree Structure. One is to study the algebraic properties of the structure. Once people realized this structure is a quite complicated one, methods from logic started to be used to show it is actually that complicated. Another approach has been studying how order-theoretic properties of certain Turing degrees relate to properties about their computational power. There is a lot of interaction between these approaches and I have been interested in this program in general. I have written a survey paper on the history of the study of the Turing Degree Structure via embeddability results where I mention my contributions to the area until 2006.

A few results

Determinacy. With Richard A. Shore, we obtained the precise limit of how much determinacy can be proved without the use of uncountable objects.

Theorem (Montalbán, Shore 2009)**.** *Given a fixed number n, second-order arithmetic can prove that every Boolean combination of n $\mathbf{G}_{\delta\sigma}$-sets is determined. However, for each n a different proof is needed and no single proof works for all n: second-order arithmetic cannot show that all Boolean combinations of any number of $\mathbf{G}_{\delta\sigma}$-sets are determined.*

Consider a set A of binary sequences, or equivalently, a subset of Cantor set. Players I and II play binary bits alternatively for infinitely many turns, forming an infinite binary sequence; player I wins if the sequence is in A, and, otherwise, player II wins. We say that A is *determined* if one of the players has a winning strategy for this game. Statements about determinacy of games have attracted logicians for many decades because of the high complexity of the winning strategies, and also because they have been a useful combinatorial tool in a wide range of areas. Martin's celebrated theorem says whenever A is a Borel subset of Cantor set, A is determined. But, as noticed by Friedman, Martin's proof requires unusually big cardinals almost never used in mathematics.

Second-order arithmetic, a widely studied logical system, captures all mathematics that can be done in the realm of countable objects. Countable objects include real numbers, continuous functions, real analysis, countable algebra, separable metric spaces, etc.. Almost all of mathematics that can be modeled with, or coded by, countable objects can be carried out in second-order arithmetic. So, our theorem finds exactly how much determinacy can be proved using only countable objects.

A subset of Cantor space is $\mathbf{G}_{\delta\sigma}$ if it is the countable union of countable intersections of open sets (i.e. it is in the third level of the Borel hierarchy); a set is a *Boolean combination* of n sets $A_1, ..., A_n$ if it can be defined form them using (finite) intersections, unions and complements.

The study of the determinacy of $\mathbf{G}_{\delta\sigma}$ sets seemed intractable for a long time, and so did the study of results at the boundary of second-order arithmetic. Our result broke those two barriers.

Computable presentations of structures. In 1955, Clifford Spector proved that every hyperarithmetic well ordering is isomorphic to a computable one. In less technical terms this says that if an ordinal has a representation of a certain complexity (hyperarithmetic, which is quite high) then it also has a very simple (computable) representation. This theorem is central in Hyperarithmetic theory. I proved the following surprising generalization to all countable linear orderings:

Theorem (Montalbán 2005)**.** *Every hyperarithmetic linear ordering is equimorphic with a computable one. (Two linear orderings are* equimorphic *if they can be embedded in each other.)*

The proof of this Theorem requires a deep analysis of the structure of the countable linear orderings modulo equimorphisms. This analysis led me to define equimorphism invariants for the class of scattered linear ordering of any size. These invariants are finite trees whose nodes are labeled by ordinals. They are equimorphism invariants in the sense that two linear orderings are equimorphic if and only if they are assigned the same invariant. The invariants provide a new description for the partial orderings induced by the embeddability relation on the class of scattered (and of all countable) linear orderings. I have written a survey paper about my results on linear orderings before 2006.

Linear orderings. Fraïssé's conjecture (proved by Laver in 1971) is the statement that says that the countable linear orderings form a wqo with respect to embeddablity. It has interested logicians for many years because of the difficulty of its proof in terms of reverse mathematics; it uses constructions which are computationally more complicated than most of the theorems of mathematics. From my work, it follows that this statement has a *robustness property* in the sense that it is equivalent to many other statements talking about the same type of objects. It also follows that to assume Fraïssé's conjecture is sufficient and necessary to develop a reasonable theory linear orderings and the embeddability relation. So far, the only systems with this robustness property were the main five, but we do not know if Fraïssé's conjecture is equivalent to one of these five. It was conjectured by Clote in 1990 that it is equivalent to Friedman's system of Arithmetic Transfinite Recursion (ATR_0). This problem is still open and plan to work on it. One possible approach is to study the length of the wqo's involved, since this usually gives proof-theoretic information. Together with Marcone we have recently made some progress on this approach; we calculated the length of the wqo of linear orderings of finite Hausdorff rank, obtaining proof theoretic consequences, plus of course, a better understanding of the structure.

Boolean Algebras. Studying the definable relations on a structure is an important theme all throughout logic, and for the computability view point, it is useful to understand how many Turing jumps it takes to compute a certain relation. With K. Harris we obtained a complete characterization of the relations on a Boolean algebra that are defined within a certain number of Turing jumps.

Theorem (Harris, Montalbán 2007)**.** *For every n, there is a* finite *complete set of computably infinitary Π_n formulas for the class of Boolean algebras.*

The relations definable withing n jumps are exactly those definable by computably infinitary Π_n formulas, and a set is complete for this kind of formulas if it captures all the structural information that can be captured by these formulas. Most classes of structures do not have natural complete set of Π^c_n formulas, and even less finite ones. Researchers have been looking for a result of this sort for some time, as it is the first step to solve the well-known open question of whether every low_n Boolean algebra has a computable isomorphic copy. We achieved our characterization by getting a very good understanding of the back-and-forth relations on Boolean algebras, and we believe this is a key step towards the solution of this open problem, and will also be a useful tool for other work on Boolean Algebras. These work has already been useful for us to build a low_5 Boolean algebra not $0^{(7)}$-isomorphic to any computable one, and has been used by Harris to characterize the relatively arithmetically categorical Boolean algebras.

Turing Degrees. The Turing degrees form an *upper semilattice*; that is, every pair of elements $\mathbf{a}$, $\mathbf{b}$ has a least upper bound $\mathbf{a} \vee \mathbf{b}$. Intuitively, $\mathbf{a} \vee \mathbf{b}$ contains all the information that $\mathbf{a}$ and $\mathbf{b}$ have toghether. There is another naturally defined operation called the *Turing jump* (or just *jump*). The jump of a degree $\mathbf{a}$, denoted $\mathbf{a}'$, is given by the degree of the *Halting Problem* relativized to some set in $\mathbf{a}$. (Given $A \subseteq I\!N$, the *Halting Problem relative to* A, denoted by A', is the set of the (codes for) computer programs that, when run with oracle A, halt.) It can be shown that the jump operation is strictly increasing (i.e., $\forall \mathbf{a}(\mathbf{a} <_T \mathbf{a}')$) and monotonic (i.e., $\mathbf{a} <_T \mathbf{b} \implies \mathbf{a}' <_T \mathbf{a}'$). A *jump upper semilattice* is an upper semilattice together with a strictly increasing, monotonic function.

One approach to understanding the shape of the Turing Degree Structure has been by studying the structures that can be embedded into it. Kleene and Post, in the same paper where they introduced the Turing degree structure in 1954, proved that every finite upper semilattice can be embedded into $(\mathbf{D}, \leq_T)$. Since then, various other embeddablity results have been proved. For countable structures, the most general result proved so far is the following:

Theorem (Montalbán 2003)**.** *Every countable jump upper semilattice can be embedded into the Turing Degrees $(\mathbf{D}, \leq_T, \vee, ')$ (of course, preserving join and jump).*

Another interesting result I proved along these lines is that the question of whether it is possible to embed every jump upper semilattice of size $\aleph_1$ satisfying the countable predecessor property into $(\mathbf{D}, \leq_T, \vee, ')$ is independent of ZFC.