

## Research Statement

ANTONIO MONTALBÁN

February 8, 2010

My area of research centers around Computability Theory, a branch of Mathematical Logic. The general program that is behind most of my research is the one of measuring the complexity of mathematical practice. That is, studying and understanding the complexity intrinsic in mathematical objects, statements, constructions and proofs. The complexity of objects is usually measured using the notion of “computable from”. Studying the relation of “computable from” abstractly, disregarding the meaning of the objects being computed, is part of Pure Computability Theory, an area where I have also done work.

I have work in many areas inside computability theory, and even some outside computability theory, but in this short research statement I will restrict to comment on the programs and problems I have dedicated most of my time. I will start by briefly describing the basic ideas in general terms, and then I will describe a few of my results.

### BASIC CONCEPTS

The main concept in computability theory is the relation “computable from”. A set  $A \subseteq \mathcal{N}$  is said to be *computable from* a set  $B \subseteq \mathcal{N}$ , 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  $\mathcal{N}$  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 way of doing the coding is usually obvious and inessential. Here is an example: we can encode the set of finite triangulations of simply connected compact manifolds with a set of natural numbers. It can then be shown that this set 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 encodes the set of group presentations given by (generators, relations) representing a non-trivial group (called the *word problem*).

**Computable Mathematics.** This area of computability theory is concerned with the computable aspects of mathematical objects and constructions. When we look at structures (like rings, fields, orderings, graphs, etc.), we ask questions like the following: Can information be encoded into an isomorphism type of a structure? When can a mathematical structure be represented computably? Or, in other words, what algebraic properties guarantee that a structure can be represented computably or not? More generally, how do algebraic properties of a structure interact with its computational properties? When we look at mathematical constructions, the first question one asks is whether it can be carried out computably, and if not, what oracle is necessary to perform the construction?

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 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. Since explaining this would require more background, I will describe other, more concrete, aspects of my work in computable mathematics below.

**Reverse Mathematics.** The questions of which axioms are necessary to do mathematics is of great importance in Foundations of Mathematics and is the main question behind the program of Reverse Mathematics. Reverse Mathematics deals with the system of second-order-arithmetic which is rich enough to be able to express an important fragment of classical mathematics. This fragment includes number theory, calculus, countable algebra, real and complex analysis, differential equations, separable metric spaces and combinatorics among others. Almost all of mathematics that can be modeled with, or coded by, countable objects can be done in second-order arithmetic.

The idea of Reverse Mathematics goes as follows. We start by fixing a basic system of axioms. The most commonly used basic system is called  $\text{RCA}_0$  that essentially says that computable sets exist. 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 a very interesting fact is that most of the theorems, whose proof-theoretic strength has been analyzed, have been proved equivalent over  $\text{RCA}_0$  to one of five systems, that we will call the *main five systems*. Understanding why only these main five systems are so frequently equivalent to theorems of ordinary mathematics is a very intriguing question.

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.

**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 Turing 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  $(\mathbf{D}, \leq_T)$ .

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 together. 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  $A \subseteq \mathbb{N}$ , 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.,  $\mathbf{a} <_T \mathbf{b} \implies \mathbf{a}' <_T \mathbf{b}'$ ). A *jump upper semilattice* is an upper semilattice together with a strictly increasing, monotonic function.

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 algebraic properties of certain Turing degrees in this structure 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.

## SOME RESULTS

**Determinacy.** Statements about determinacy of games have attracted logicians for many decades because of the high complexity of the winning strategies for games that are not that complex. (And also because it has been a useful combinatorial tool in many proofs.) Consider a set  $A$  of sequences of natural numbers. We define a game  $G(A)$  played as follows. Players I and II play natural numbers alternatively for infinitely many turns, forming an infinite sequence of natural numbers; player I wins if the sequence is in  $A$ , and otherwise, player II wins. We say that  $G(A)$ , or just that  $A$ , is *determined* if one of the players has a winning strategy for  $G(A)$ . (A *winning strategy* is just a function that tells you what to play given the moves played so far, and such that if you follow it, you always win.) Not every game  $G(A)$  is determined, but most of the games we might encounter are. The space of sequences of natural numbers has a natural topology given by the product topology of the discrete topology on  $\mathbb{N}$ . It requires a simple proof to show that if  $A$  is open, then  $G(A)$  is determined. A much more complicated proof is needed to show that every

Borel set is determined. The axiom of choice is necessary to build games that are not determined. Second order arithmetic (also called analysis) is an important logical system where most of mathematics can be carried on. However, it is known that much less than Borel determinacy can be proved there. With Richard A. Shore, we obtained the precise limit of how much determinacy can be proved in second-order-arithmetic. To state our result, we need a couple definitions: A set of sequences 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  $A_1, \dots, A_n$  if it can be defined from them using (finite) intersections, unions and complements.

**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.*

**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 with nodes labeled with 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.

**Boolean Algebras.** One of the questions I am the most interested in solving is the following well-known open question in Computable Mathematics.

**Q:** Does every  $\text{low}_n$  Boolean algebra have a computable isomorphic copy?  
(A set  $X$  is  $\text{low}_n$  if it has the same  $n$ th Turing jump as the empty set.) It has been proved that every  $\text{low}_4$  Boolean algebra has a computable copy, but the general case remains open. With K. Harris we have made considerable progress on the question. We achieved a very good understanding of the back-and-forth relations, and we believe this is a key step towards the solution of the problem, and will be a useful tool for other work with Boolean Algebras. I have also started to studying similar behaviors on other structures and analyzing structural reasons for that behavior.

**Length of well-quasi-orderings.** Here is a typical example of the effective-content analysis of a theorem: DeJongh and Parik 1977 proved that every wqo has a linearization that is maximal among all its possible linearizations. (A *well-quasi-ordering*, or *wqo*, is a quasi-ordering without infinite descending sequences and without infinite antichains.) The order type of these maximal linearizations, called the *length* of the wqo, is often used to measure well-quasi-orderness when used in applications in combinatorics, proof theory and re-writing systems in Computer Science. So far there was no uniform procedure to find maximal linearizations, and every proof computing them used some new idea or trick. I have shown that this can be done computably; every computable wqo has a computable linearization. However, it cannot be done uniformly, that is, there is no single computer program that is able to find these maximal linearizations, assuming a program for the computable wqo is given as input (there is not even a hyperarithmetic procedure). My interest in understanding the computability aspect of this procedure came from the study of Fraïssé's conjecture (see below) and other results from Proof Theory.

**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 embeddability. It has interested logicians for many years because of the difficulty of its proof in terms of reverse mathematics; it uses constructions which are more computationally 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 ( $\text{ATR}_0$ ). 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.

**Turing Degrees.** 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 embeddability 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*.