IEEE 4th Symp. on Logic in CS '89

[Sco89]

DOMAINS AND LOGICS

extended abstract

Dana S. Scott
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This year marks twenty years since domains and domain-theoretic models for the λ-calculus were discovered in the fall of 1969 while I was working with Christopher Strachey at Oxford on the semantics of programming languages. As I have tried to explain many times, I was trying then to convince Strachey *not* to use the untyped λ-calculus, since I felt (rightly, I still believe) that at the time it was just a formal theory without strong connections to standard logic and mathematics. What Strachey needed was a method of flexible function definition—including definition by recursion. He also wanted to treat procedures as first-class objects—at least in some languages. Finally, there had to be a way of proving properties of programs.

The approach I was trying to sell Strachey was based on what I felt were well understood ideas from recursive function theory using higher-type functionals. These ideas came from work by Kleene, Kreisel, Myhill, Shepherdson, Davis, Nerode, and Platek—to name a few. The notion of continuity—for certain kinds of functionals—had been clearly emphasized by Kleene and the Fixed-Point Theorem was used by Kleene and much earlier by Tarski (from whom I learned it) to express recursive definitions. These were my starting points.

What I going to give Strachey (which I did in an unpublished manuscript) was a simple type hierarchy of functionals starting with the partial functions from numbers to numbers at the bottom and working up the types by taking only the continuous functions at each stage. ("Continuous" implies "monotone", and this was also known to many of the above workers in recursive function theory.) Continuity was justified—as Kleene had himself indicated—because it meant that, roughly speaking, a finite amount of information about the value of a function is already determined by a finite amount of information about the argument. This is very much to the point when you think about how these functionals can be computed.

The Kleene school, however, had not done extensive development of the theory of partial functionals for what I believe are two reasons: they were interested in total functions and they were interested in the recursion-theoretic and descriptive-set-theoretic properties of functionals that are not continuous (such as quantifiers over infinite sets). Kreisel and Kleene had discussed the "countable" functionals, but again they were concerned with the case of the total functionals. Kreisel's work was important for me, however, in that he had spent much time on describing the topology of his spaces of functionals, and I worked on the other theory of partial functionals partly by analogy.

One strong motivation for working out this theory in just this way for Strachey was the success I had had earlier in the summer of 1969 in joint work with Jaco de Bakker on the semantics of program schemes. The use of continuity and fixed points was central here, and the difference in the new work was the passage to higher types.

My first original discovery, I believe, was that in the theory of partial functionals (what turns out to be continuous functions on algebraic cpo's), each type has a countable basis for its topology. (That is to say, the continuous function space between countably based algebraic cpo's is again countably based.) I was able to see this by connecting up the topology with the basis of finite elements of the space. (Note that these elements do not occur in the spaces of total functionals.) In particular, I gave a combinorial description (since much refined by many people) of how the bases at lower types are fitted together to give the basis at the higher type. When you see how to do this, you

CH2753-2/89/0000/0004$01.00 © 1989 IEEE

see explicitly why everything stays countable for basis elements.

My next discovery (November 1969) was based on the following reasoning: If ordinary spaces of functionals have modestly complicated bases, and the complication multiplies when you pass to the function space, then could there not be a space D with already such a complicated basis that when you formed the continuous function space from D into D the resulting basis was isomorphic to that of D? I did not particularly want to find such a space at the time, but I had to follow up the idea to see if this were possible. You cannot have a clear conscience in Mathematics if you do not follow up the possibilities. I did not at that time give the combinatorial proof of the existence of this basis, but instead I used the idea of inverse limit from topology to put higher-type spaces into a sequence with a limit space. This was easier to do than giving a complete recipe for the final basis—though a complete description of the bases of reflexive spaces obtained this wa is easily possible.

The success of this method then led me to think about the kind of spaces obtained and about how to have a general theory. Many discussions with Strachey, Park, Reynolds, and others, led to my introduction of domain equations—that is to say, the recursive definition of types. It was somewhat later in response to questions from Robin Milner, that I formulated the definitions in terms of a calculus of retracts. And it was later still in response to a report written by Plotkin that I suggested how to use universal models and the language of recursive definitions that goes along with them both for functions and for domains through retracts.

At nearly the same time, Ershov in Russia was working out his theory of f-spaces also motivated by the theory of functionals in recursive function theory. He discovered the connections with topology that I also had to find out for myself and he related the whole theory to effective computations. The Ershov school did not discover the reflexive spaces and models of the λ-calculus, however.

This brief recap of the beginnings does not begin to do justice to the previous work of many people, and it does not even hint at the subsequent developments in domain theory over the last twenty years. The lecture will cover some of the later history and will be principally aimed at answering several questions:

(1) How successful is the theory of domains in giving a model theory for the semantics of programming languages?

(2) What has happened to the program of proving properties of programs? Has Domain theory made this possible?

(3) Do domains give us sufficiently many types? Or do we need more extensive theories?

(4) Does the whole notion of domain make better sense by allowing a change in foundations from classical logic to a constructive logic well adapted to properties of computable functions?

I personally believe that the answers to these questions are not yet definite. The reason for giving the lecture is to discuss the issues critically, particularly as regards Question (4).