Just a caveat

With X and Y denoting arbitrary types, we assume the availability of the following functions:

$$\begin{array}{lcl} b & : & X \to Bool \\ t & : & X \to Nat \\ f & : & X \to Y \\ g & : & X \to X \\ \oplus & : & X \times Y \to Y \end{array}$$

Suppose that b, t, and g satisfy

(0) $(\forall x : x \in X : \; b.x \Rightarrow t.(g.x) < t.x)$.

Then we have the following

theorem: A unique function $F : X \to Y$ exists satisfying

(1) $$(\forall x : x \in X : \; F.x = \begin{array}{l} \mathbf{if} \; \neg b.x \to f.x \\ [\!] \;\; b.x \to x \oplus F.(g.x) \\ \mathbf{fi} \end{array})$$

proof: By straightforward mathematical induction on the value of function t, with use of (0) of course.
□

Type X may have a rich algebraical structure; yet, in general F, as defined by formula (1), is not a homomorphism (or: catamorphism) on X.

The so-called Unique Extension Property from Category Theory is not applicable to definition (1) ; yet, the theorem states that (1) uniquely defines a function $F : X \to Y$, and its proof is simple.

By the above I do not intend to belittle the current research on a category theorical approach to programming — a large class of problems lends itself very well for this approach — , but I do wish to point out that category theory is not the whole story. The above example illustrates this ; a more "realistic" example of the same phenomenon is the well-known merge sort.

Eindhoven, 23 october 1991
Rob R. Hoogerwoord