The motivation for this paper is that of providing a full model of the language Quest (see [4]), a very rich programming language involving impredicative type quantifiers and subtyping within three levels of kinds, types and type operators (extending $F\omega$) together with recursive types and fixpoint operators. A complete answer to the problem of providing a model for Quest has not been given yet. There are various partial answers: in [2] Amadio makes the first attempt to deal with impredicative type quantifiers and recursive types, and in [1] Abadi and Plotkin give a Partial Equivalence Relation description of essentially the same model as Amadio’s, gaining the possibility of using a categorically sound environment for maps between types, but they leave unanswered the question of how to combine recursive types and bounded quantifiers. In [4] Cardelli and Longo develop a very different PER model which is suitable for the part of Quest which does not involve recursive types. To summarize, the problem is that of finding a well-behaved subcategory of PER’s where all objects have sups of chains and a bottom element, and all maps are continuous.

The way to develop it is that suggested by Dana Scott in [22] and pursued in [21]: to use the logical properties of an appropriate topos to describe the categorical model as if its objects were sets. Indeed our choice shall go to some very peculiar sets which have sups of chains by their own nature. We shall take complete extensional PER’s including the empty (=bottom) function in a realizability topos: the notion of impredicative type quantification will be the standard, set-theoretic one for these cases, the notion of recursive type will be given by the usual inverse limit construction, and that of subtype will be intended as {-,-}-closed subsets.

The use of category theory, and of topos theory in particular, for the study of models of strong computational languages has undergone a heavy propaganda in recent years. The fundamental argument which is put forward is that it simplifies many proofs, and gives an intuition of the object at study which is closer to its proposed computational meaning. In the first cases, simplifying meant making arguments actually easy; we do not claim that it does so in this paper. PER models indeed seem to have been one of the motivating ideas in structuring Quest, but then the syntactic complexity of the language has removed it from the ideal model. Indeed, after giving a fairly intuitive interpretation for Quest, we can pinpoint some aspects of the semantics which do not agree with the intended syntactic meaning, and we suggest a different interpretation which solves some of the problems. In particular, this new interpretation allows recursive operators to act on all types.

The major aspect in the intuition of a type in a language like Quest is algebraicity, in the sense that a type should be thought of as determined by the algebraic properties which
are requested for it. Although this is a common request in programming languages with a type discipline, it is the more felt in the case of Quest, as it presents kind constructors which display a natural dependence on their type variables. So in an intended semantics, the interpretation of a type should be taken up to isomorphism. The reader must be warned not to think of an interpretation of a type as the isomorphism class of an object. The new interpretation of a type will be (something like) an object with all its possible isomorphic representations. This overcomes the problem of subtype reindexing: in the interpretation of subtypes as inclusions of PER's, given a function/term \( f: S \to T \) and a subtype \( c_{X,T}: X \to T \), one cannot expect in general a subtype \( c_{Y,S}: Y \to S \) defined by “inverse image”. From this, various problems ensue, notably the impossibility to extend the recursion operator to all types. The idea of an interpretation where types are taken up to isomorphism was brought up by a number of people in Cambridge already in '86, and it occurs in [12], [5], but nobody had pointed it out clearly until it was revealed in [18].

In section 1 we review very quickly the logical properties of a realizability universe and recall the completeness results about modest sets which are necessary for developing our intentions. In section 2 we recall the essential definitions of extensional PER’s, and state the crucial results about them which enable us to give the model. In section 3 we extend our work to give the new categorical interpretation suitable to model Quest with recursive operators on all types. In section 4 we suggest how to extend our work to get categorical interpretations of a similar flavour to that of Abadi and Plotkin’s.

1 The basics

It is by now natural to embed a PER model in an appropriate realizability topos. The way to do this is given in [11] as follows. Suppose \( A \) is one of the following instances of a partial combinatory algebra: the standard applicative structure on the natural numbers \( N \) given by a coding of the partial recursive functions, the \( \lambda \)-model \( P\omega \), the initial solution to the domain equation \( D = (D \times D) + ([D \to D] + N + \{\bot, \ast\}) \). (Most of what we say hold in wider generality than these three cases, but it is not relevant for our purposes.) The topos \( \mathcal{A} \) is defined introducing a logic calculus based on the proposition-as-types paradigm: the elements of \( A \) are to be thought of as codes for proofs. So the logical values are the subsets of \( A \) and the logical operators act in a constructive manner, for instance the conjunction of two subsets \( P, Q \subseteq A \) is the set \( \{ (p, q) : p \in P, q \in Q \} \) for \( (, ) \) a fixed surjective pairing function on \( A \). And the implication \( P \Rightarrow Q \) is the set \( \{ a : \forall p \in P.a \cdot p \in Q \} \).

The objects of the topos are pairs \( X = (|X|, =_X) \) consisting of a set \( |X| \) and a \( P(A) \)-valued equality \( =_X : X \times X \to P(A) \), in the sense of the logic calculus previously mentioned: there are \( a, b \in A \) such that for every \( x, y, z \in |X| \)

\[
\begin{align*}
a \in [x =_X y \Rightarrow y =_X x] & \quad \text{and} \quad b \in [x =_X y \land y =_X z \Rightarrow x =_X z] \\
\end{align*}
\]

respectively. Maps are defined following the same intuition, as \( P(A) \)-valued relations which are single-valued and everywhere defined. For complete details, we refer the reader to [11]. The important fact is that the definitions above describe a topos \( \mathcal{A} \), and the category of PER's on \( A \) is isomorphic to a full subcategory of \( \mathcal{A} \), an excellent reference for this is [9]. Moreover, it is the global sections of an internal category of \( \mathcal{A} \)... Here the subject becomes more complicated, and categorical logic offers an easy way out.