A Complete Coinductive Logical System for Bisimulation Equivalence on Circular Objects*

Marina Lenisa

Laboratory for the Foundations of Computer Science
University of Edinburgh, Scotland.
lenisa@dcs.ed.ac.uk

In recent years, considerable energy has been devoted towards the development of simple principles and techniques for understanding, defining and reasoning on infinite and circular objects, such as streams, exact reals, processes, and other *lazy data types*. Structural induction trivially fails on infinite and non-wellfounded objects. It can be applied only indirectly, using particular implementations of these objects, e.g. streams as inductively defined functions on natural numbers. Elaborate mathematical theories, such as domain theory and metric semantics, can be used, of course, to support rigorous treatment of such objects. But an ideal framework should allow to deal with infinite computational objects in a *natural*, *operationally based*, *implementation-independent* way, without requiring any heavy mathematical overhead.

Systems based on *coinductive definitions* and *coinduction proof principles* appear to be a good starting point for developing such a framework. These are natural, in that infinite or circular objects and concepts often arise in connection with a *maximal fixed point* construction of some kind and, moreover, they can be justified often simply by elementary set-theoretical means, see e.g. [5]. In many situations, simple categorical concepts, such as those of Final Semantics ([1, 7, 5]), are enough to achieve a substantial generality. In this context infinite objects are described as terms of F-coalgebras for suitable functors F's. The Final Semantics paradigm has been proved to be a quite powerful technique for giving final semantics to programming languages. A wide spectrum of languages, which comprises *process algebras*, *imperative concurrent languages*, *functional languages*, *π-calculus*, and various equivalences thereon, have been modeled in the Final Semantics setting (see [5]).

In this talk, I will present a formal *logical* system for reasoning on infinite objects, which generalizes [2] and it is inspired by the seminal work of Coquand [3] (see also [4]) in the Intuitionistic Type Theory setting. In particular, I will present a system *à la* Gentzen $\mathcal{S}_{co}$ for capturing bisimulation equivalences on circular *regular* objects, i.e. objects which have only a *finite* number of non-isomorphic subobjects. The language for describing objects is typed. Types are defined as follows

$$\sigma \ ::= \ X \ \mid \ K_1 \ \mid \ldots \mid \ K_n \ \mid \ \sigma + \sigma \ \mid \ \sigma \times \sigma \ \mid \ \mathcal{P}_f(\sigma) \ \mid \ \nu X.\sigma \ ,$$

where the coinductive type $\nu X.\sigma$ is guarded, i.e. all the free occurrences of the variable X in σ are within the scope of a type constructor, and $\mathcal{P}_f(\ )$ denotes the finite powerset type.

The set of terms is defined as follows:

$$t \ ::= \ x \ \mid \ c^i_j \ \mid \ i_1(t) \ \mid \ i_2(t) \ \mid < t, t > \mid \ [t, \ldots, t] \ \mid \ rec\, x.t \ \mid \ in(t) \ ,$$

where $\{C_j \equiv \{c^i_j \mid i \in I_j\}\}_{j \leq n}$ are sets of constants, $[\ldots]$ denotes the multiset term-constructor, and the other constructors are standard. The term $rec\, x.t$ is guarded, i.e. all the free occurrences of the variable x in t are within the scope of one of the following term constructors: $i_1(\ )$, $i_2(\ )$, $<\ ,\ >$, $[\ldots]$.

Many infinite objetcs usually dealt with in lazy programming, such as processes and streams, can be easily seen to have a formal counterpart in our typed language.

The crucial ingredient in the formulation of our logical system is the following *coinduct-*

* Work supported by HCM contract "Lambda Calcul Typé", Esprit Working Group "Types", MURST ex 40%.

ive rule: $\dfrac{<\Delta; \Gamma, in(t) \sim in(t') : \nu X.\sigma >\vdash_{co} t \sim t' : \sigma[\nu X.\sigma/X]}{<\Delta; \Gamma >\vdash_{co} in(t) \sim in(t') : \nu X.\sigma}$ (in) . Notice that the conclusion can be used as auxiliary hypothesis in establishing the premise. In a sense, our system can be viewed as a system for infinitely regressive proofs, inspired by the *guarded induction principle* of [3].

In order to give external independent justifications to our system, we consider two equivalent semantics. The first is a fixed point bisimulation semantics, the latter is based on Final Semantics. We show that our system is both correct and complete w.r.t. them. The fixed point semantics is defined by introducing, for each type σ, a bisimulation equivalence $\approx_\sigma$ on the set T^0_σ of closed terms typable with σ. This family of equivalences is defined as the greatest fixed point of a monotone operator. In order to give the categorical semantics, we define a "universal" functor F, involving constructors corresponding to each of the type constructors. Then we endow the family of closed typable terms $\{T^0_\sigma\}_{\sigma \in Type^0}$ with a structure of F-coalgebra, in such a way that the greatest F-bisimulation on the coalgebra of terms coincides with the family of bisimulation equivalences $\{\approx_\sigma\}_{\sigma \in Type^0}$. This yields a final semantics for our typed language, with respect to which the system $\mathcal{S}_{co}$ is complete. For simplicitly, we work in the category Set^* of *non-wellfounded sets* and set-theoretic functions.

Our system, when restricted to the type of CCS-like processes, amounts to a logical system for establishing strong equivalence of processes, alternative to the axiomatic system of Milner, [6]. The crucial difference between $\mathcal{S}_{co}$ and Milner's system is that Milner's rule for recursion, *viz*: $\dfrac{t \sim t'[t/x]}{t \sim rec\, x.t'}$, is replaced by rule (in) above.

There are various interesting directions for possible generalizations and extensions of the coinductive axiomatization presented in this talk. First of all, we could generalize our language also to non-regular objects, still getting a sound axiomatization. A further possible generalization is that of considering a larger class of types, including inductive types and the *mixed* covariant-contravariant $\rightarrow$ constructor. Another possible generalization is that of axiomatizing notions of bisimulations other than Milner's strong bisimulation, like, for example, weak bisimulation and congruence, van Glabbeek-Weijland branching bisimulation, etc., possibly aiming at a general presentation of all these branching-like bisimulations.

References

1. P.Aczel, N.Mendler. A Final Coalgebra Theorem, in *Category Theory and Computer Science*, D.H.Pitt et al. eds., Springer LNCS **389**, 1989, 357–365.
2. M.Brandt, F.Henglein. *Coinductive axiomatization of recursive type equality and subtyping*, *TLCA'97* Conference Proceedings, P.de Groote, R.Hindley, eds., Springer LNCS **1210**, 1997.
3. T.Coquand. Infinite Objects in Type Theory, Types for Proofs and Programs *TYPES-93*, Springer LNCS **806**, 1994, 62–78.
4. E.Giménez. Codifying guarded definitions with recursive schemes, *Workshop on Types for Proofs and Programs*, P.Dybjer et al. eds., Springer LNCS **996**, Berlin 1994, 39–59.
5. M.Lenisa. Themes in Final Semantics, Ph.D. Thesis **TD-6/98**, Dipartimento di Informatica, Università di Pisa, March 1998.
6. R.Milner. A complete inference system for a class of regular behaviours, *J. of Computer and System Sciences* **28**, 1984, 39–466.
7. J.J.M.M.Rutten, D.Turi. On the Foundations of Final Semantics: Non-Standard Sets, Metric Spaces, Partial Orders, *REX* Conference Proceedings, J.de Bakker et al. eds., Springer LNCS **666**, 1993, 477–530.