Logic, resource, and reflection

L.G. Meredith[2]
and Michael Stay[1]

[1] Pyrofex Corp.
stay@pyrofex.net
[2] RChain Cooperative
lgreg.meredith@rchain.coop

Abstract. We present an algorithm for deriving a spatial-behavioral type system and term assignment from a formal presentation of a interactive computational calculus. This turns out to identify a species of category which we offer an axiomatic characterization of interaction categories.

1 Introduction and motivation

In groundbreaking work, Abramsky, Gay, and Nagarajan put forward the idea of interaction categories to give a categorical framework for interactive models of computing. Indeed, interactive models of computation are relatively new and offer distinct insights because they include the computational environment as part of the computation. Examples include the π-calculus, in fact, all the mobile process calculi, as well as the lambda calculus. However, interaction categories fail to say axiomatically what interaction is.

We put forward a simple, intuitive axiomatic characterization of interaction and use this to derive not only a logic but a proof theory and term assignment algorithm for all systems satisfying these axioms. The resulting structure identifies a species of category and we offer this as an axiomatic characterization of interaction categories.

1.1 Intuitions

The key idea in this construction is to use both the evaluation context of Meredith and Stay, and computational reflection, similar to what is found in Meredith and Radestock, to build a proof theory with a cut elimination that corresponds exactly with the notion of computation embodied in any interactive rewrite system.

The requirement that the rewrite system is interactive means the left hand side of every rewrite rule will necessarily be a term constructor, say K, taking at least two terms. That is, when viewed as a piece of syntax it is at least a tuple of terms. Then, by controlling evaluation with an evaluation context we can force a distinction between a tensor and a cut, where the cut forms the redex with an evaluation context supplied, and the tensor is denied the evaluation context,

effectively rendering it a data structure. Of course, the data is inaccessible until there is a means to unpack it, and a par term is constructed as the principal means to extract the data from the tensor.

The use of the term constructor that forms a redex as a data structure is the essential use of reflection. The lack of a reduction context gives us the ability to suspend computation. Having suspended it, we reify computation as data, and use par to unpack the data making up the computation. Completing the circle, the tensor-par cut rule constitutes the means to reflect computational data back into actual computations.

It turns out that denying the evaluation context is not the only way to suspend computation. We can also form contexts. In this case, we only admit 1-holed contexts, ranged over by χ. With contexts we can define a notion of rely-guarantee, two notions actually (indicated by the operations $(- \rhd -)$, and $(- \lhd -)$), as a redex term constructor might not be commutative. .

$$\tau \rhd_{\mathsf{K}} \tau' \triangleq \{u \quad | \quad \exists t.u = !\mathsf{K}[]r(t,[]), \forall u' : \tau.(\exists \rho : u@u' \to v) \Rightarrow v : \tau'\}$$
$$\tau' \lhd_{\mathsf{K}} \tau \triangleq \{t \quad | \quad \exists u.t = \mathsf{K}[]l([],u)!, \forall t' : \tau.(\exists \rho : t@t' \to v) \Rightarrow v : \tau'\}$$

In the sequel we drop the subscript on the triangles if it's understood from context. We use the notation $\chi@t$ to mean the term formed by substituting t for hole in χ. Thus, $\mathsf{K}(t,[])@u = \mathsf{K}(t,[])[u/[]] = \mathsf{K}(t,u)$ and similarly, $\mathsf{K}([],u)@t = \mathsf{K}([],u)[t/[]] = \mathsf{K}(t,u)$

To give some examples, the comm rule of rho-calculus is given by

$$\mathsf{for}(y \leftarrow x)P \quad | \quad x!(Q) \to P\{@Q/y\}$$

In this case K is parallel composition, and the revised, resource constrained comm rule, looks like

$$R|\mathsf{for}(y \leftarrow x)P \quad | \quad x!(Q) \to P\{@Q/y\}$$

In the lambda calculus, β-reduction is given by

$$(\lambda x.M)N \to M\{N/x\}$$

K is application, and the revised, resource constrained β-reduction is given by

$$R((\lambda x.M)N) \to M\{N/x\}$$

In what follows we will focus on calculi that don't employ binding operators and so-called nominal phenomena. This is not to say that we can't handle nominal phenomena, just that the content is already complex enough and we want to focus on the core ideas.