The rho calculus 20 years on

L.G. Meredith[1]

CEO, F1R3FLY.io 9336 California Ave SW, Seattle, WA 98103, USA,
lgreg.meredith@gmail.com

Abstract. We discuss developments and advances in the rho calculus 20 years after its inception.

1 Introduction

This December will find my submission of the first paper about the rho calculus to be 20 years in the rear view mirror. A lot has happened since then. Apart from production implementations of scalable decentralized asset management systems based on the calculus, a lot of work has been done on the rho calculus. Indeed, just this May (2024) i discovered a conservative extension of the calculus that fills a hole in the account of reflection that has long bothered me. And, last April (2023) i found a fuzzy version of the rho calculus that is just waiting for someone to take further. In 2018 i discovered a variant of the calculus (which i dubbed the *space calculus*) that makes it a computational analog (in a certain precise sense) of solutions of the Einstein field equations.

Thankfully, i am not the only one who has found the rho calculus interesting enough to think about. Stay and Williams studied a variant they called the π-rho calculus as an example of a system that could be typed in their native types construction. And ??? showed that rho calculus is more expressive than the the π-calculus. The list goes on.

The aim of this note is to collect and summarize in one place developments over the last 20 in the study of the rho calculus. i suspect that there are more researchers than i know about who are interested in the questions raised by the calculus. Hopefully, this note can serve that community, and possibly spark wider interest due to the discoveries to date.

1.1 Summary of contributions and outline of paper

More specifically, this note will develop the following.

- the modern, programming-language-friendly syntax for the calculus;
- a handful of useful syntactic sugar coatings that make the calculus a suitable semantic framework for a realistic protocol-oriented programming language, including, but not limited to:
  - adding first class values;
  - adding unforgeable names (the π-rho calculus);
  - adding syntax for joins and sequences and choice;
- the fuzzy variant of rho;
- the space calculus;

– probabilistic and quantum variations of rho;
– revisiting the encodings of π into rho and vice versa.

But, we will also touch briefly on the techniques used to represent reflective computation in the rho calculus and demonstrate that they are not bound to this specific setting. In particular, they can be applied the the λ-calculus and to set theory, yielding variations of these theories that have very useful features.

Finally, we will conclude with some thoughts about the foundations of mathematics and its relationship to agency. As a preview, we believe that set theory (and even to some extent category theory) are theories of *data structures*. Implicitly the community has believed that mathematicians have agency and their formalisms need only carry information content between these agents. But the advances in artificial intelligence (AI) tell a different story. To some extent AI has brought their formalisms to life and is on the verge of giving them agency. As a result, i want to argue that a proper foundation of mathematics needs to include a notion of agency and that the ingredients of a foundational account of agency is present in the mobile process calculi, generally, and a pragmatic account is present in the rho calculus, specifically.

2 Concurrent process calculi and spatial logics

In the last thirty years the process calculi have matured into a remarkably powerful analytic tool for reasoning about concurrent and distributed systems. Process-calculus-based algebraical specification of processes began with Milner's Calculus for Communicating Systems (CCS) [20] and Hoare's Communicating Sequential Processes (CSP) [13], and continue through the development of the so-called mobile process calculi, e.g. Milner, Parrow and Walker's π-calculus [23], [24], Cardelli and Caires's spatial logic [4] [6] [7], or Meredith and Radestock's reflective calculi [18] [17]. The process-calculus-based algebraical specification of processes has expanded its scope of applicability to include the specification, analysis, simulation and execution of processes in domains such as:

– telecommunications, networking, security and application level protocols [1] [2] [14] [15];
– programming language semantics and design [14] [11] [10] [31];
– webservices [14] [15] [16];
– blockchain [19]
– and biological systems [8] [9] [26] [25].

Among the many reasons for the continued success of this approach are two central points. First, the process algebras provide a compositional approach to the specification, analysis and execution of concurrent and distributed systems. Owing to Milner's original insights into computation as interaction [21], the process calculi are so organized that the behavior —the semantics— of a system may be composed from the behavior of its components. This means that specifications can be constructed in terms of components —without a global view of the system— and assembled into increasingly complete descriptions.