The Faraday Form Index

Summer 2022, Portland State University
Justin T. Chun ($f\mathbf{G}i$), Liubomir Chiriac (PDX)

ANSI Common Lisp, described most succinctly as the "programmable programming language," is most useful in creating domain-specific languages (DSLs) that wrap around conventional programming languages in order to take advantage of the most advanced metaprogramming facility extant. When one understands the scope of Lisp, any language becomes tractable (and hence intuitively comprehensible), freeing the programmer from the birdcage of hyperspecialization.

Seasoned Lisp hackers are, arguably, on an entirely different plane of agency than every other type of programmer currently out there. So it is only natural that it be a Lisp hacker who finds its successor. If Lisp is essentially the initial object of expression, what would the corresponding final object look like? Computation being the teleological enterprise it is, such an object would resemble an ordinal, perhaps Cantor's absolute infinity $\mathbf{\Omega}$.

Many new programming languages attempt to democratize Lisp by softening the edges. The most successful attempt to date is Ruby, the Latin to Lisp's Ancient Greek. Whither does Sanskrit lie? Along the tangent bundle of the dyson sphere, and yet coincident with the floor of the abyss. Recent attempts to improve upon Lisp most notably feature the `sectorlisp` initiative. This is impressive and meaningful work, but it is essentially an incremental improvement. The advent of `sectorlisp` has sparked a renaissance in Lisp thinking, leading its ambassadors to dream up a yet catchier slogan: the Maxwell's equations of software.

Hang on, though. Maxwell's equations essentially frame electromagnetism through a scalar field (electricity) and its corresponding 1-form (magnetism). Very Greek indeed. One can think of Greek as having a French-esque syntax with German semantics. On the other hand, Sanskrit wields German syntax alongside French semantics. Thus, if Lisp commands finance, the language tantamount to the Faraday 2-form (henceforth `Faraday`) ought to moderate media. This is genuine novel development: one can reckon Lisp macros to Durer's engravings, then ask how one might sculpt a tool that is flexible and accomodating enough to create Monet sunrises.

What's more flexible and accomodating than mathematics itself? If one were to create a bespoke DSL, with all the proper tools, the resulting amalgam would revolutionize the pace, cadence, and momentum of novel mathematical research. Well, there's no need to be subjunctive about all this: such components already exist. On the proving algebraically side, we have agda for motivoids and pari-gp for automorphs. If we zap the abyss floor with a powerful enough laser, the scintillation would rock the dyson sphere like a live-wire Faraday cage.

`sectorlisp` and G. Spencer Brown's calculus of indication give us the German syntax we need to generalize as specifically as desired. Whence do we find our wings, then? Well, for starters, there should be no real distinction between function, macro, and special form. Within `Faraday`, everything should be a form.

An inside source has informed the authors that end-to-end lossless graphics is on the horizon for hardware. This means raytracers like PoV-Ray and 2D tools such as asymptote and postscript can now be utilized to their true capacity. With `Faraday`, the grab-bag of curated forms arising in frontier mathematics research, one could sketch out the idea on a agda/pari/gp repl, then debug geometrically by crafting forms that dictate the structure and values of the output space.

In other words, Lisp and `Faraday` are the derived functors of code and data, respectively. Yin and Yang.

Portland, Oregon
29 November 2021