POINT OF VIEW

William Bricken
August 1994

Here's my point-of-view on POV:

POV has been cleanly excised from scientific thought and notation for hundreds of years. It is not surprising that it turns out to be difficult to understand. It is also fundamental to any deep understanding of our work to be able to understand and incorporate pov.

Boundary math (bmath) is particularly designed to incorporate pov into mathematical thought and notation. (This does not imply that it does it well.) Pov is assumed in all notations, most usually we view from outside the notation. VR is an exception which provides viewing from inside. Bmath provides the tools to make pov explicit.

Divergence

From a purist perspective, to understand pov, one needs to simulate/incorporate the bmath axioms into their mental models and experience. This is basically the route that I took. Like teaching, writing, and several other wide-spread activities, folks think that skill is natural, inborn. Wrong! Skill at all levels takes years of conscientious practice. Thus I find it difficult to place credibility in opinions about the foundations of bmath and pov which do not have a basis in conscientious learning. In particular, bmath has Platonic structure, it is not a collection of opinions. In cognitive science, the human route to knowledge goes through

exposure -> recognition -> evaluation -> application -> understanding

If you find yourself in the opinionated-evaluation stage, you are least qualified to render a credible opinion. Folks often say "convince me", I say "leap of faith". To be more clear, the leap is not one of abandonment of sense, it is one of willingness to put in the years of study without falling into premature evaluation. I add this divergence because, if we are to understand pov, we must be willing to do pov exercises and studies. This requires, paradoxically?, abandoning one's pov. That is, understanding pov precludes opinions about pov. This is hard for almost everyone.

Convergence

I spent about five years thinking about the cognitive implementation of crossing and calling. About two years in, the idea of "thinking about"

degenerates into the idea of "implementation". In another year or so, "implementation" degenerates into "incorporation". The path is similar to attainment of any expertise, going through conscious exercise to semi-conscious practice to unconscious skill.

So what do

( ) ( ) = ( )

and

(( )) =

mean in cognitive terms?

"=" means "is confused with". You can't tell the difference.

To confuse ( ) ( ) with ( ), you need to be able to collapse multiplicity in your perspective, to see One world, One people, One thought. Incidentally, this is the function of a mantra. Operationally, it means to maintain only one thought. We have probably all approached this during deep concentration on a task.

So, to know calling, practice until you can hold one idea in your mind without fluctuation, for say 10 seconds.

When you do this, it is easy to see that

(.) ( ) = ( ) (.)

that there is absolutely no difference between the apparent two. So we write:

(.)

There are also simple simulation exercises. Consider (.) to represent a room with you in it. In the simulated world, all rooms are indistinguishable. When you look around the room, you do not see any other empty room. The ( ) structure is purely imaginary. When you change rooms, you cannot tell a difference, so you have no way to verify that indeed you have changed rooms. Changing rooms is thus an imaginary act. The imaginary room and the imaginary act are accessible only from an outside pov. Even then they are purely imaginary distinctions. And as is the case for all bmath, the distinction between object and action is also imaginary. But we learn that there is an asymmetry: imaginary constructions are discernable from outside one's pov, but not from inside. Lots of famous philosophy has been written as to why this happens to be the case. The origin of these exercises is the Leibniz-Clarke correspondence in the late 17th century, where Clarke was

actually representing Newton. Leibniz called it The Principle of the Indiscernible.

To confuse (( )) with   , you need to be able to collapse pov itself, to see No world, No people, No thought. Incidentally, this is classical enlightenment, and is reached through the internalization of the mantra. Operationally, it means to maintain no thoughts or images. So, to know crossing, practice until you can (not)hold no thoughts, for perhaps 10 seconds.

When you do this, it is easy to understand why

(.( )) =

and why there is no accompanying simulation exercise.

Here's a model I have grown to like:

Calling is like having two doors into a room. Functionally, you can use either, it is equivalent to having one door. Crossing is like going through a door and going through it again. Operationally, it is equivalent to not having moved. Calling is like having two locks on a door, using either is equivalent. Crossing is like locking and unlocking a lock, the state of the door does not change.

The above pov exercises immediately generalize to an algebra (they must, since you are free to carry any pov):

A ( ) = ( )

(A ( )) =

Finally here are some implementation consequences of generalizing pov:

In my distinction networks paper, I specify how the implementation of dnets fundamentally changes when using an interior pov (compared to the LISP exterior pov we currently have). Behavior of the system also changes, but the result of operations does not. To me this says that graph implementations are not canonical, and can only be seen to be equivalent from a black-box perspective.

I/O, and directionality in general, implicates the selection of a pov. I had a difficult time understanding circuits which are completely closed (no i, no o), but now believe that is the only appropriate stance for analysis. That is, any I or O represents the imposition of a pov, and is thus a special case. This leads into seeing circuits as closed systems, and introduces the rich field of general systems theory into circuit analysis. Or: we should

be able to cut a closed circuit at any point to construct a pov. This seems like what is done in probe analysis.

Hierarchy then collapses to another pov operator, from the outside rather than inside. So pov operations are all we are doing:

hierarchy:	. A	outside
i/o:	(. A)	inside
initialization:	.	boundary