Name Spaces and other Musings

Michael P. Frank
Symbolic Systems Program
Stanford University

June 12, 1991

1 Introduction

One day last March, I was visiting Carnegie-Mellon University, and I stopped by the library's Computer Science Reading Room to take a break from the rigors of grad-school recruitment. I browsed a bit, and settled down with a Usenix journal, wanting to see what their articles were like. There was an article proposing a system for file naming in Unix that resolved logical pathnames into user-defined directories. This got me thinking about all the different name spaces in Unix, and so this paper reproduces and cleans up the notes I took in the reading room on the musings that followed.

2 Numerous Naming Sytems

Modern computers use a multitude of different *naming systems*, or ways of mapping from some sort of symbolic name to its referent. Just of the top of my head, here are some of the naming systems I know of that are in use on the Unix computers I use every day:

- An *Internet address* is mapped by a *name server* to an *IP address* which is mapped by a *router* to a *physical machine.*
- An *environment variable* is mapped through a *process environment* to a *string.*
- A *shell variable or alias* is mapped through the *shell bindings* to a *string.*
- A *pathname* is mapped by a *filesystem, network filesystem, and symbolic links* to a *file inode.*
- A *command* is mapped through the *command path* to a *file.*

- A *virtual address* is mapped by the *page table* to a *location in RAM or disk.*

- A *tilde username* is mapped through the *password file* to a *home directory pathname.*

- A *object symbol* is mapped by the *linker* to a *library function.*

- A *dynamic variable* is mapped thorugh a *language's binding system* to its *value.*

Notice that these naming systems operate on many different levels of the computer's functional hierarchy, from memory cells to machines on a network. Why are there so many different levels of naming systems? Partly because of historical development. But it results in an overly complex system to learn, with all these different naming conventions.

The point of all naming is to get from *less information* to *more information*, from a short name to a long binding. But why do we bother with the *less information* version? Because *less information* is easier and faster to deal with. Naming is possible, or makes sense, because there are certain contexts where you don't care about some information, but do care about certain others, so you assign the *less information* names to reference the things you do care about.

Sometimes naming is a process, a relatively simple piece of code that accesses a (possibly larger) piece of information.

Again, why is naming so complex? Because of historical development, and the permanence of standards. For example, Lisp alone has a simpler and more coherent naming system than Unix alone, but Lisp these days tends to be embedded under Unix, and so it is forced to deal with Unix's naming system (file pathnames). But even Lisp has a complex and heterogeneous naming scheme (resulting from its historical development): packages.

With all this revealed, can we develop theory that can be the basis of new computer systems that are much simpler, but equally powerful? Probably, but it would never be a "standard." It could be educational, though.

The most finely-grained level that can be a context for naming is this: an individual process on an individual machine, and moreover, a single function within that process.

Note htat even if small pieces of information can only be individually named by names that are longer than the small piece of information (e.g., byte addresses), the existence of a regular relationship between many small pieces (e.g., an ordering) and a corresponding computable relationship between their names (e.g., incrementing) allows large amounts (many pieces) of information to be accessible given something shorter – the name of one piece, together with a real or implied computation (e.g., looping through data) which implies a corresponding structure among the many pieces.

Indeed, one can argue that the information necessarily takes form in the *structure* of how the pieces are put together; disjoint scattered bits have little informational value; a structure adds information. (Of course, even structured bits, though they carry information in a technical sense, still have no meaning, in themselves, objectively; that requires a context, or rather, and interpreter, be it a program or a person.)

So information requires an arrangement of bits, and most of the information only comes in when you add the structure. Could there be, perhaps, a way in thinking about information in which *all* of the information is in the structure, and there are no fundamental "seeds" of information that the structure ties together? For it seems odd that there should be a necessary duality of structure-and-bits when neither, alone, can usefully embody information. The would is not cleanly divided up into bits of information.

So how should we study naming?

- Asynchronous flow of bits down a wire and back, at different rates?
- Mapping from bit-strings to other bit-strings?

Both of these are biased towards totally ordered sequential structure on the information. Is that necessarily a "correct" structure? On what basis? It corresponds to our notion of time. Ultimately, information output is a time-series flow, or at least that's all we can think of.

The idea of computing is that a relatively low rate of information flow into the computer system (from the user, say) is interpreted by the system to produce a relatively high rate of information output (to the user) which may then be interpreted and trigger still more interaction. However, the output need not be immediate; it's not when the user is entering a program (or data) to be used later. Some of the user's input may be just to elicit an informational response. Some may be to configure the future behavior of the system. Some may be both. There's a shading between the two. [1] But basically, all communication into a computer (or human) is for the purpose of being interpreted by the receiving entity, and possibly shaping the nature of its future ways of interpreting. All communication within a computer, in any of the various naming systems, networks, programs, can be seen in the same way: as a mutual molding of ways of interpretation, a mutual structural coupling. Of course, this division of the world into shaper and shapee (and particular instances of such) is merely a way of seeing the situation.

In any case, let us return to the question, What is naming? A "naming scheme" is a standard *way* of modifying how future interactions will be interpreted. Of course, in setting up a naming scheme, we are are also doing this, in a way specified by the creator of the underlying system. Every stage of the interaction between human and computer, from designing the chips to clicking

[1] Communication between computers, as well as between computers and humans, has these same properties

the mouse, is a shaping or limiting of future possibilities for interaction. The same as with the education of a human.

Why limit future possibilities? (1) We have to. Everything we do limits future possibilities, by increasing entropy, and making a decision that cannot be changed. There is a deep connection between ever-increasing entropy and ever-more-finely-shaped spaces of possibilities. (2) Entities with more limitations stand out more. We notice them more. They have more of character of their own, more of a structure. They carry more information. They are interesting. And in the presence of energy flow, they can survive and grow. This is life. There is a deep connection between loading a register and life. (3) By limiting future possibilites, you make things possible that otherwise would not have been possible. By having a word processor on your hard disk, you make it possible to write a letter in a hurry, though you have limited the space of possibilities for what else may be on the disk.

So what's the practical consequence of all this? How does philosophical understanding of what it means for entities to interact have any bearing on the here and now, on the design of our computing systems? So whatt if everything we do is shaping future possibilities? Sure, modern computer systems have layer after layer of different naming conventions for different things. But that's what makes them interesting. In the end, is their usefulness hurt by the multiplicity of different interfaces, languages, and protocols? As they say, "The great thing about standards is that there are so many to choose from."[2] And to use in conjunction with each other. The computer is an interlocking network of different dependencies, and it's impossible (or at least very difficult) for one person to understand them all, but as long as we can get our work done, we are happy. Some of us. Others find their joy in the shaping of possibilities, the toolmaking. We shape the world (our conceptual systems included) far more than other animals.

The computer is a shaped tool that affords many new possibilities for its further shaping. Is is a minimally-shaped tool. Perhaps that's what's so different about it? Why it's better than other tools? Why customizable programs are better than non-customizable ones? Musings: certain people like customizability. And computers. Customizable things afford more more opportunities for shaping. They do less shaping themselves. Shaping and being shaped. It's a delicate balance. The computer is able to shape and be shaped, making it superior to machines that go one way or raw material that goes the other way.[3] People are able to shape and be shaped, making us better than lower animals, or raw matter. Better? Superior? Well, at least more adaptable, a better survivor.

So what is AI trying to do, after all? It is a shaping, a shaping of computers to shape and be shaped more like the we ourselves shape and are shaped by the world around us. Through AI we are shaping the way we live and work in

[2] At least, Carl Witty says this.

[3] Computers are both machine and raw material.

the future, as with all tool-shaping. The computer is especially alluring because we can form a tight symbiosis with it, mutually shaping each other's spaces of possibilities.[4] Why do we want to create AI? So that the mutual shaping will be even tighter? Hm. We want the computer to speak our language, for that would make it easier for people to enter into the mutual shaping—they would have to be shaped less themselves.

But as always, the question becomes, just how are *we* shaped? What is our makeup that lets us shape the world the way we do, and be shaped by the world the way we are?[5] "Know thyself" says the ancient maxim. And is it necessary to have a machine shaped like ourselves? We want to know if we can do it. It's a matter of species pride. But for usefulness, experience seems to show that shapability is key, not excellent mimicry of some particular shape (e.g., ours). I wonder... Why does "limiting possibilities" help at all? Because it brings other possibilities closer to hand, or makes them present. The nice thing about human minds is that they can temporarily limit possibilities, bringing some possibilities closer to hand, and yet if it doesn't work out we are able to throw away those limitations (somewhat) and "pop up" a level or two, approaching the situation again with an open mind (although there is undoubtedly *some* lasting limitation, i.e., learning). Computers, too, can make assumptions and later reject them if they didn't work out. That's not the problem. Humans seem to know which assumptions to reject (ideally, at least, we do) and they can reject quite a lot of them. And yet, we don't explicitly assume them! We just can't write down the enormous bulk of assumptions, limitations, that we are capable of rejecting. So how do we see that we've been making the assumption, how do we reject it? More intelligent humans seem to have more ability to see and question their assumptions. How?? How do you recognize the presence of an assumption in your thinking if you didn't already explicitly represent it? You need to reflect on your thinking, analyze it to figure out what the assumptions are, using some other assumptions that form the space of possibilities for what your assumptions might be like! Form an explanation for your thinking, figure out a hypothesis for what-it-is-you're-doing, and then explicitly do a negation of that. Still unclear. Given some pattern of reasoning, a proof or argument, and then a revelation about what implicit assumption was there, do some learning, so you can guess in the future about what assumptions to identify and then relax.

So... say you have a program (or generally, a computer system) that does something (anything). It can be regular or AI-oriented. So now: by observing the system, and its interaction with other systems, can one figure out what the implicit assumptions are? And when the system is guessed to be breaking down, can one take the corrective measures to loosen some of the assumptions? That's easy if the assumptions are isolated into an explicit, descriptive element, hard if they're not. How about the brain? Once a connection is wired in can it be

[4] Reference Maturana

[5] *Bias* is another name for the initial shaping that affords certain kinds of further shaping. Other words that have been used in this way: *background*, *prejudice*, *program*.

undone? Or, can all repair be done by layering more stuff on top?

So, can one create an assumption-relealing program? Where will it sit? It will have to build in additional assumptions, itself. One can only identify a *shape*, a restriction to a space of possibilities, from the perspective of some larger space of possibilities. E.g. the idea of revealing assumptions seems to imply some representation for those assumptions, and thus assumption-descriptions are selected from some space which allows certain kinds. (And you can't escape by allowing "multiple representation schemes", for such scheme-descriptions themselves are described in some space.) Why do biological-type systems seem to avoid this restriction? Why do they seem able, under environmental pressures, to change in totally unforeseen ways? Unrepresented ways? One thing is this: biological representations and structures seem adaptable to new semantic values. Thus, a limited space of possible structures may have a nearly unlimited range of possible roles in interaction with the medium. The actual use of the structure is determined, though structural coupling, to be the one appropriate to the context.[6] This ties in with our reasoning by analogy. An assumption may be revealed through drawing an analogy between one conceptual structure and another, thus giving and old structure new semantic value. But how do we know how to try to rebind the semantics of something? How do we figure out what structural element combines with what semantic element? Incremental trial and error and repair? So, ideally we can analogize anything with anything else. And more intelligent, or more creative people do this better than less creative individuals. Analogizing is rather like chromosomal crossover in the wealth of new ideas it produces. Evolution works through creating analogies? Ditto for thinking? Emphasis on "structures." What are they? Relational, semantic net of conecepts? "X activates Y" kind of thing? And an analogy is a built-up series of cross-connections between concept structures, a whole set of cross-connections activated all at once. The analogy itself is a concept. Very Minskyan.

3 Summary of Important Points

[6]Total reference to Maturana here.