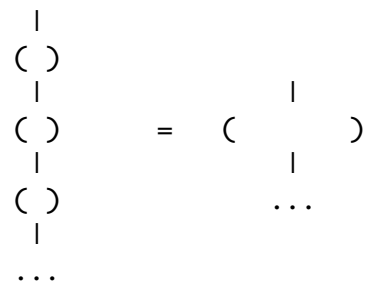
THE SECRET OF ORGANIZATION

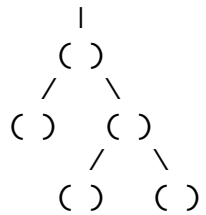
William Bricken
April 1989

Object hierarchies that are thoughtfully constructed can be normalized into different forms. We recognize the simple case of this with a string of inheriting objects. We can choose to compress any portion of the string into a single object. Diagrammatically:



Branches

It's the branches that define the structure of the organization we are calling an object hierarchy. Branches tell where and in what ways subclasses diverge. It's the difference that makes the difference.



Constants as Grounds

Every hierarchy has a bottom. In tree structures, the bottoms are called leaves. In algebraic structures, the bottoms are called constants. In object-oriented implementations, bottoms are run-time instances. All constant grounds are trivial to absorb into the operators in their context, since the operator is just a map (a table-lookup) over constants. So

```
3 + 4 ==> 7
```

because the left-hand-side (lhs) is just a syntactic variation of the right-hand-side (rhs).

```
foo + 4 ==> foo +4
```

The only difference between the lhs (operating on a constant and an unknown) and the rhs (doing a slightly different operation on the unknown) is when the work is done. That is:

Constant Absorption is a lazy dynamic compiling technique.

The result of this observation is that Constants do not need to be the grounds. They can be seen as function modifiers. In general:

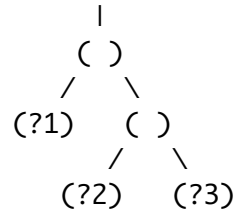
F[..., constant, ...] = Fconstant[...]

Variables as Grounds

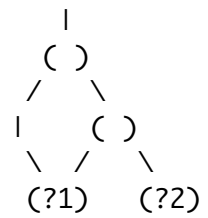
Variables represent what we don't know. The purpose of computation is to remove doubt by evaluating variables. Often the problem is just a twisted expression. Someone said something poorly. For instance, were I to say "Here's A and B and A and B and A and ...", you would say that I am being redundant, saying something poorly. I've got a case of the dreaded infinite loop.

More often the twisting of the expression is because what we know (the input) is messy. The person who drew the room in AutoCAD didn't say what was inside and what was outside. (The Open Plan drawing really is open.) The information needed to treat the room as a 3D object can be supplied by using Variables in place of knowns. This technique is called Generalization, or Abstraction. It is stepping back from instances (1,2,3) to possibilities (a,b,c). Since things like a chair in Open Plan might have known edges, the Open Plan is a mixed model. We can *figure out automatically* the missing edges of the room by mapping the known portion onto a rectangular parallelopiped, unmixing the model. (90% of architecture maps onto rectangular parallelopipeds, presumably because that's the abstract model architects use for abstract visualization.) We know the map visually, of course, and the strange name is totally internal to the mathematical rather than the conceptual model. The concept Room includes the generic default shape Block.

So Variables are the ground of computation. Its where the buck stops, where the value returns. Lacking value, the Variable is a Literal, it is what it is, a name of something we don't yet know.



The critical observation here is that *each variable must be in a single location*. Tree structures fail to be efficient cause the forced binding of a variable on the far left branch is not immediately conveyed to other occurrences of that variable, say on the far right branch. The key is to move from a Tree Hierarchy to a Constrained Graph Hierarchy. The constraint is that only Variables have multiple upper neighbors. With this structure, the flow of control can "teleport" out of a goal-driven local process into an information-driven process in a totally different locality.



Transformations of the Network between Us and the Grounds
are the same as
Message and Object Hierarchies between Us and the Grounds

The Secret of Organization is to build a network of components within which Branches indicate Differences. The Network places a Structure between Us and our Ignorance.