Using 3D to Optimize Computations

Lewey Geselowitz, July 4th 2021
Published by lewcid.com at
http://www.lewcid.com/lg/articles/using-3d-to-optimize-computation.pdf

Summary: this article describes my primary mental approach or technique for programming in which the elements of memory, information and time are charted spatially, in a manner that allows complete accuracy in representing the original computation. I have used this mental visualization almost every workday for the past 12 years since discovery to untangle and optimize innumerous complex software processes into more efficient or useful configurations. It takes some getting used to, thinking of software as a shape, but is provably complete, shows a route towards optimality, and allows the visual cortex of the brain to assist in the act of writing software.

Computation can be seen as the process of combining the information within memories into actionable signals or states. The optimization of a computation is the creation of a new process which achieves an equivalent resulting signal or state using fewer computational resources (I.e. generally less memory or time), often in the context of one or more reference performances or simulations of the original computation. We endeavor to show the utility towards finding these optimized sub-processes achieved via the automateable examination of the constituent computational elements in past performances. For optimization and visualization purposes, the original reference computational performances can be (Turing-)completely expressed in a binary spatial modeling language, or “shape” using these fundamental three input and one output dimensions:

	Usual Axis	***Dimension***	***Domain***	***Representation***
in	x	Memory	space-like	object { }
in	y	Time	time-like	array [ ]
in	z	Value	information-like	number / string / pointer
out	a	State	opacity-like	Opaque when memory-value-time intersect; clear where they are known to not intersect, and semi-transparent where not known.

Figure 0: Table of axes used in Turing-complete spatial visualization of computation.

Comprehensively expressing a computation as a physical shape provides a visual language of introspection useful in optimization; as the essential process steps when traced back from a given output from an exact subset of the computations volumetric manifold (I.e. the optimized process is essentially a sub-shape of the original computation). Most importantly, using an information-like dimension for value rearranges the constituent signals from memory-space into usage-space, connecting copies and combinations independent of time or location, a vital step

in identifying mathematical optimal routes of information within merely forward time computational flows. The size of this volumetric expression is upper bounded at O(n^3) of the number of elements in the computation, being the cubic intersection of all possible memory addresses, times, and value strings; making optimization a non-trivially complex yet finite process itself, particularly suitable to focused application on critical sub-processes whose aggregate and sometimes continuously repeated resource cost often justifies the finite investment in optimization. Eventually the automated identification and optimization of software processes while in operation could achieve self-optimizing computational systems, a generalized from of a computation-aware caching system, but until then this technique has benefits as a mental approach to modeling the where, when and what of a software process as a visually tangible and flexible conceptual construct. With appropriately scoped application, this process of optimization via spatial untangling, can thus be useful for those who optimize or modify software on a regular basis. It certainly has been for me,

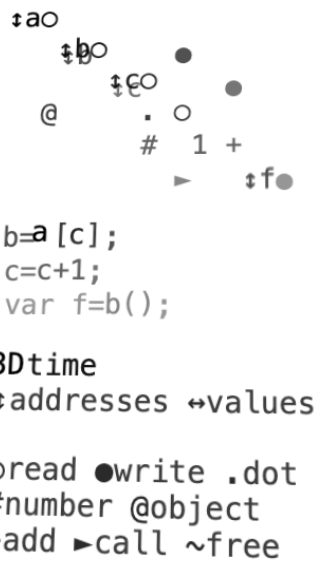

Figure 1: Still image from 3D visualization of computational fundamentals.
Realtime version here: https://leweyg.github.io/lewdo/index.html?code

-Lewey Geselowitz @ Lewcid.com