Kenneth Jahnke

Lab 2 - 02 Feb 2025 (due date extension granted)

CSCI 364 (Online) - Spring 2025

Crucial to my understanding of this paper was recognizing that it was originally written in 1968. I, a modern programmer using modern programming languages, wouldn't think to use explicit `go to` statements because modern flow control structures largely manage the state of a program on my behalf - I merely define what states are significant.

Dijkstra's example of tracking the `n` number of people in the room was a good example of why the use of `go to` statements for tracking the progress of a process is not ideal. Firstly, the use of `go to` in such a process is tedious and inefficient. Manually defining where and when within the processes to update the value of `n` dampens the value of using a machine for this task. (Such a procedure is more akin to notching a tally on physical notepad than utilizing a sophisticated piece of technology.) This is the point he was making in his plea to "shorten the conceptual gap between the static program and the dynamic process...." A machine given a proper flow-control mechanism will automatically track `n` more efficiently and very likely with less error than a series of manual `go to` statements. If a machine is to be used, capitalize on its capabilities. Secondly, use of `go to` statements for management of minutia (such as the update of `n - 1` to `n`) draws attention away from the utility of a program and onto its code-level implementation (the details of which hold little meaning). Heavy use of `go to` statements creates a "bureaucracy" in manipulating the value of `n` that detracts from core objective of the program. Thirdly, a "rogue" `go to` statement could be injected to divert the processing locus to a point irrelevant to the current subprocess. (Historically speaking, this is not only a waste of time, but of money too as users had to pay a fee for access to computers of the time.) At best, this is a distraction from the objective of the process. At worst, it creates a nightmare situation for debugging.