



MPT Kernels

The Core Concepts of Blue Cheetah CUB

Author

Kevin D. Howard

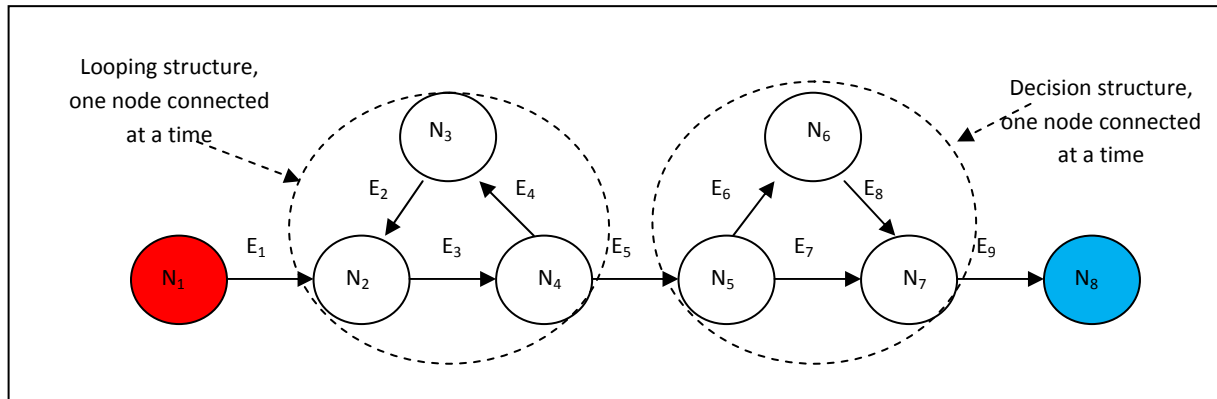
Original Date: 20 September 2010

Update: 29 September 2010

In computer science, the term “kernel” usually refers to the most essential functionality of an operating system. In mathematics, the term typically refers to certain key mapping functions. Recently, the Defense Advanced Research Projects Agency (DARPA) has identified a kernel as a small self-contained software code used to compare computer architectures. The salient feature of these definitions is the concept of essential functionality. Massively Parallel Technologies, Inc., (MPT) has determined that the essential functionality of all algorithms occurs when a data transformational process has no embedded control, that is, when the data transformational process is stripped of looping, decision-making, or subroutine-calling structures. In addition to essential functionality, we are left with the essential control structure. When only these two component types are left, we have decomposed the algorithm into its MPT Kernels: MPT Process Kernels (MPKs) and MPT Control Kernels (MCKs). An algorithm that has been decomposed into its constituent MPT kernels is called an MPT Algorithm Decomposition (MAD). When the decomposition of an algorithm has code that is not an MPT kernel, the non-MPT kernel code components are known as mixed kernels. Algorithms that are not MADs are called simply algorithms. There are two primary benefits to producing a MAD: automated control kernel programming and simplified process kernel programming.

Automated Control Kernel Programming – Separated control takes the form of state transitions that determine the execution order and call process kernels which have no internal control. Another way of viewing this is to say that all control can be represented as an Augmented Finite State Machine (AFSM) where the states are MPKs and the state transitions are MCKs. The AFSM is augmented as there is a need to include data movement between data stores (state-like entities that send or receive data elements, not control). With the definition of states as MPKs, state transitions as MCKs, and the inclusion of data storage and retrieval, there is enough information in a MAD to sequence through the AFSM as a program (provided that the MPKs are pre-compiled and accessible to the system). In the MPT model, the creation of a MAD occurs as part of the MPT design process, and since the AFSM can be executed directly, all executable control functionality is automatically generated directly from the design activity.

Simplified Process Kernel Programming – By definition, an MPK contains no control structures. This means that the created software is linear. According to Cyclomatic Complexity (CC) analysis (developed by Thomas J. McCabe, Sr.), the complexity of a software code is a function of the count of linearly independent code segments. The higher the value generated by the analysis, the less maintainable and the more error-prone is the code. Code with a value greater than nine is not considered to be maintainable. To perform the analysis, software source code is converted into a form that can be represented as a directed graph containing the basic blocks of the program. These basic blocks are the MPT process kernels, the linearly independent code identified in CC¹.

Source Code as a Directed Graph¹

The Cyclomatic Complexity Equation is:

$$M = E - N + 2P$$

where: M = cyclomatic complexity

E = the number of edges (arrows) of the graph

N = the number of nodes (circles) of the graph

P = the maximum number of connected components (simultaneous connections per node)

For the code represented above, the number of edges = 9, the number of nodes = 8, and the maximum number of simultaneous connections per node = 1, which gives a Cyclomatic complexity of 3^1 .

Since a programmer codes only MPKs in the MPT system, it is as if the control elements (the edges, MCKs or state transitions for MPT) do not exist for the programmer. The M value for any particular node in isolation (an MPK, or a state in an AFSM, for MPT) is -1, where the number of edges = 0, the number of nodes = 1, and the maximum number of simultaneous connections per node = 0. The programming complexity of MPKs is, thus, always -1, and, further, the programming complexity of any MAD = -1. The -1 represents a pathless or 0-dimensional graph, a trivial structure, and the lowest level of process-kernel programming complexity possible, that is, the simplest possible linearly independent code.

¹ Description of Cyclomatic Complexity substantially from wikipedia: http://en.wikipedia.org/wiki/Cyclomatic_complexity