A geometric model for the laws of concurrent programming.

Tony Hoare.

Summary.

We present a discrete plane geometric model, in which the diagrams describe the traces of execution of concurrent and distributed programs. The points on the diagram stand for actions that have occurred during that execution; their two co-ordinates give a non-metric indication of the time and place of performance of the action.

Data flow between actions is modelled by labelled arrows between the points, where the labels give the value communicated. Control structure is modelled by repeatedly splitting the trace either horizontally or vertically into components. Each records the execution of a syntactic component of the program, for example, a method body or a thread. These two orientations of the split define the operations of sequential and concurrent composition of the components.

An ordering relation is defined to compare traces according to their degree of sequentiality or concurrency. Traces that are equally concurrent are congruent, and may therefore be declared to be equal. This permits proof of an associativity law and a unit law for both forms of composition. In addition, an interchange law permits concurrency to be executed by sequential interleaving.

A standard algebraic construction has been adapted by Mueller to define a program as the set of all the permitted traces of its execution. Non-deterministic choice is represented by the union of these sets. All composition operators distribute through union. In summary, the model obeys all the laws of Concurrent Kleene Algebra as codified by Struth. These laws validate the optimisations commonly applied to both concurrent and sequential programs.

The level of presentation requires from the audience only a memory of high-school algebra, geometry and logic, and an acquaintance with the basic concepts of modern programming. The suggested level of motivation is an interest in the methods of basic science as applied to software engineering.

Biography

Tony Hoare has a first degree in classical languages, literature, history, and philosophy both ancient and modern. He has professional qualifications as a Statistician and as a Russian Interpreter. But his primary interest is in philosophy, which he has sought to apply to computer programming and programming languages throughout his fifty-year academic and industrial career.