Graph-Based Operational Semantics

Nachum Dershowitz

A graph-based low-level, generic specification language for asynchronous algorithms is proposed, and a graph-based operational semantics for generic programs is elucidated. The graphs incorporate vertices for atomic events and edges indicating control flow alongside edges for data flow.

The basic atomic actions are reads, writes, tests, and choices. They are (quasi-) ordered by control edges. Data edges carry values from whence they are produced to where they are consumed. States are characterized by logical structures, so the actions refer to the current interpretations of operations in the structure. The control order of a computation is dictated by two considerations: Values must be retrieved before they can be used, and references to the same unchanged location should yield the same values.

The suggested formalism allows one to infer, not just input-output behavior nor only what events took place, but also how often and in what order they transpired. It may be applied to illuminate the different possible semantics of Dijkstra's guarded commands. In particular, there is no obligation to evaluate all guards before proceeding.

Inspiration was derived from the work of the Gilbreths, Turing, Petri, Gurevich, and many others.