HOARE78

Hoare, C.A.R., "Communications of the A.C.M., Volume 21, Number 8, August 1978, pp. 666-677. It is Hoare's opinion that input and output are basic primitives of programming and that parallel composition of communicating sequential processes is a fundamental program structuring model. As such, CSP (Communicating Sequential Processes) is a programming notation based on synchronous message passing and selective commmunications. Traditional approaches to introducing parallelism on single machines have included ideas such as multiprocessing in the operating system or hardware specifications. Recent developments in processor technology that have allowed multiprocessor machines to be made up from many individual processors suggest greater economy and performance. To be able to make maximum use of such configurations, however, requires processors which are able to communicate and synchronize with each other. Although many approaches have been suggested for these (i.e., utilization of common store, semaphores, monitors etc.), there is no basic criteria for choosing an approach for a particular application. Hoare, in this paper, specifies a single, simple solution to these problems based on: Dijkstra's guarded commands (sequential control structures), a parallel command (specifying concurrent execution of particular sequential processes), simple I/O constructs. The commands specified by Hoare are simple yet powerful, with the most important being the parallel and alternative commands. As mentioned above, the parallel command specifies concurrent execution of its constituent components. All components start simultaneously and the parallel command terminates only when all components terminate (either successfully or unsuccessfully). Each process of a parallel command must be disjoint from every other process in the command in the sense that it does not mention any variable which occurs as a target variable in any other process. This implies that any transfer of information must be accomplished via communication between the processes. The alternative command is particularly useful for concurrent processes. Specifying a list of component processes, it will allow only one to be executed and will terminate when the chosen process terminates. The decision as to which process to choose is made on the basis of which is the first to be ready (input guards are used for this: the first process which outputs to one of the listed processes in the alternative construct will cause that process to be ready for execution). If none of the processes is ready (all guards fail), the alternative construct will fail. Communication between processes becomes a very important issue when the context is many communicating processors. Hoare has chosen to implement two concise operators for input and output (? and ! respectively). The input operation normally receives information from some other process and stores it in a structure, while output does the converse. The I/O commands specify communication between two concurrently operating sequential processes; processes which typically are the constituent components of a parallel command. All communication between processes has been stipulated to occur simultaneously (i.e., no buffering). The requirement of synchronization of input and output commands means that an implementation will have to delay whichever of the two commands is ready first. The delay will end when the mate command in the other process is ready. It can be stated, without reservation, that CSP has had an enormous impact on the design of programs and programming languages. By defining the concurrency constructs simply and concisely, concurrent programs can be easily decomposed into small communicating processes. In recent years, a number of programming languages have been implemented based on CSP. Occam, for instance, has followed the ideas of CSP quite closely and has been used in the development of a VLSI chip, the Transputer (the transputer allows efficient networking of many concurrent processors). Hence, CSP was also influential in the development of the appropriate hardware Hoare indicated would be required for efficient implementation of his language. With the ideas of Hoare now being put into practice, all that remains to be seen is whether they will be utilized sufficiently to take full advantage of communicating sequential processes.

Possible bibliography entries for this article include:

1. concurrency
2.specific languages