Concurrent Programming in the Shared Dataspace Paradigm

H. Conrad Cunningham

Department of Computer Science
WASHINGTON UNIVERSITY
Saint Louis, Missouri 63130

This research is concerned with the study of concurrent programming languages which employ the shared dataspace model [4], i.e., languages in which the primary means for communication among the concurrent components is a common, content-addressable data structure called a *shared dataspace*. Such languages can bring together a variety of programming styles (e.g., synchronous and asynchronous, static and dynamic) within a unified computational framework. We are investigating appropriate language constructs, programming techniques, formal models, programming logics, and proof techniques. The main vehicle for this investigation is a language called *Swarm* [3].

Following the simple approach taken by the UNITY [1] model, Swarm is based on a small number of constructs that we believe are at the core of a large class of shared dataspace languages. The state of a Swarm program consists of a dataspace, i.e., a set of transaction statements and data tuples. Transactions specify a group of dataspace transformations that are executed concurrently.

Although actual implementations of Swarm can overlap the execution of transactions, we have found the following program execution model to be convenient. The program begins executing with the specified initial dataspace. On each execution step, a transaction is chosen nondeterministically from the dataspace and executed atomically. This selection is fair in the sense that each transaction in the dataspace will eventually be chosen. An executing transaction examines the dataspace and then, depending upon the results of the examination, can delete tuples (but not transactions) from the dataspace and insert new tuples and transactions into the dataspace. Upon a transaction's execution, it is deleted from the dataspace. Program execution continues until there are no transactions in the dataspace.

Our results to date are highly encouraging. We have defined the Swarm language and specified an operational model based on a state-transition approach [3]. We are exploring the implications of the shared dataspace approach and the Swarm language design on algorithm development and programming methodology. We are also developing an assertional programming logic and devising proof techniques appropriate for the dynamically structured Swarm language. Colleagues are investigating implementation issues and the use of the shared dataspace model as a basis for a new approach to the visualization of the dynamics of program execution [2]. Swarm is proving to be an excellent vehicle for investigation of the shared dataspace model.

References

[1] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley, Reading, Massachusetts, 1988.

[2] G.-C. Roman and K. C. Cox. Declarative visualization in the shared dataspace paradigm. In *Proceedings of the 11th International Conference on Software Engineering*, pages 34–43. IEEE, May 1989.

[3] G.-C. Roman and H. C. Cunningham. A shared dataspace model of concurrency—Language and programming implications. In *Proceedings of the 9th International Conference on Distributed Computing Systems*, pages 270–279. IEEE, June 1989.

[4] G.-C. Roman, H. C. Cunningham, and M. E. Ehlers. A shared dataspace language supporting large-scale concurrency. In *Proceedings of the 8th International Conference on Distributed Computing Systems*, pages 265–272. IEEE, June 1988.