IMPLEMENTING ATOMIC ACTIONS ON DECENTRALIZED DATA
(Extended Abstract)

David P. Reed
Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, Massachusetts 02139

In this paper, a new approach to coordinating accesses to shared data objects is described. We have observed that synchronization of accesses to shared data and recovering the state of such data in the case of failures are really two sides of the same problem--implementing atomic actions on multiple data items. We describe a mechanism that solves both problems simultaneously in a way that is compatible with requirements of decentralized systems. In particular, the correct construction of a new atomic action can be done without knowledge of all other atomic actions in the system that might execute concurrently. Further, the mechanisms degrade gracefully if parts of the system fail--only those atomic actions that require resources in failed parts of the system are prevented from executing.

The research reported here was begun with the intention of discovering methods for combining programmed actions on data at multiple decentralized computers into coherent actions forming a part of a distributed application program. The primary concerns were that it be easy to coordinate such combined actions with other concurrent actions accessing the same data, and that it be easy to handle failures in any part of the combined action.

In the course of the research it became clear that coordinating access to data and recovery from failures were complementary mechanisms aimed at achieving the same goal--providing data and program modules whose behavior is easily specified without consideration of the details of the choice of data representation or the sequence of primitive steps executed that achieve the behavior. This goal is the familiar "information-hiding principle" elucidated by Parnas [1]. Atomic actions form one such class of modules. In this paper, we describe a new method for synchronization and failure recovery that fits well into a decentralized system. We concentrate here particularly on the application of this method to the implementation of atomic actions.

In the full paper, we first define atomic actions and discuss the problems of implementing atomic actions in a decentralized system. We then describe a way of thinking about objects as sequences of unchangable versions that reflect the sequence of changes made to the object over time. Updating an object will be thought of as creating a new version, while reading an object will be thought of as selecting the proper version and obtaining its value. As part of this discussion, we describe two complementary techniques for coordinating the versions of multiple objects--*possibilities*, which are groups of tentative versions created by updates that can be "simultaneously" installed, and *pseudo-times*, which are used to select the versions of objects that are accessed by a particular program. We then discuss how programs executing in a distributed system use pseudo-time and possibilities to access objects, and in particular, how atomic actions are constructed. An example illustrating how the techniques work is then described. Finally, we compare these methods with traditional synchronization and recovery mechanisms.

This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract number N00014-75-C-0661.

References

[1] Parnas, D.L. On the criteria to be used in decomposing systems into modules. *Comm. ACM* 15, 12 (December 1972), 1053-1058.