ON SYNCHRONIZATION AND ITS SPECIFICATION

Krithivasan Ramamritham  
Robert M. Keller

Department of Computer Science  
University of Utah  
Salt Lake City, Utah 84112

ABSTRACT

We introduce a very high level language for specifying synchronization properties. It is designed using the primitives of temporal logic which facilitates the specification of both invariant and time-dependent properties. The paper begins with a discussion of properties that affect synchronization. The specification language then introduces features constructs to express each of these in a fairly natural and modular fashion. Since the statements in the language have intuitive interpretations, specifications are humanly readable. Also, since they possess appropriate formal semantics, unambiguous specifications result.

INTRODUCTION

Processes executing in parallel interact with each other either through messages or by sharing resources whose states they modify. When message passing is the means of interaction, one process sends a message to another and synchronization of the receiving process occurs through the message [6]. On the other hand, processes that communicate by sharing resources do so by executing specific operations which change the state of the resource. Execution of these operations is synchronized by some appropriate mechanism e.g., monitors [5]. As a result, the processes that share the resource are also synchronized. In this paper we address issues related to the latter paradigm of interaction.

Various mechanisms have been proposed to synchronize parallel processes, each introduced to correct perceived deficiencies in their predecessors. Also, languages exist for specifying synchronization. Many of them, however, have made expressing synchronization problems and constructing their solutions quite difficult.

We approach the problem of synchronization at a level close to a human conceptual model of such problems. We first discuss properties that affect synchronization of concurrent processes and then proceed to introduce a specification language that facilitates expressing these properties. Evaluation of the language features precedes concluding remarks on our approach to specification of synchronization.

---

1This material is based upon work supported by the National Science Foundation under Grant MCS-77-09369 A01.
First we explain our concept of "synchronization". When concurrent processes access a shared resource, these accesses need to be controlled in such a way that each access leaves the resource in a consistent state. We will refer to the controller of accesses as the synchronizer and the act of controlling as synchronization. For correct interaction, each distinct shared resource must have a synchronizer. Access to a shared resource is through specific operations, the execution of which is controlled by the synchronizer. Constraints essential for maintaining the integrity of the resource are enforced by the synchronizer.

Concurrent processes can access a shared resource by requesting execution of any of the operations allowed on the resource. A request is serviced by the synchronizer after ensuring that none of the constraints is violated. A serviced request becomes active when it is executed either by the synchronizer or, on its behalf, by another process. We further assume that

1. There may be a finite delay between servicing a request and its subsequent activation.

2. An active process cannot be aborted or interrupted.

3. An operation remains active for a finite but indefinite period of time, after which it is said to have terminated.

The properties of a shared resource and the processes sharing the resource are discussed below.

**Mutual Exclusion of Resource Access** Very often, when a process modifies the state of a resource, it should be provided exclusive access. For instance, a write access to a disk file should be permitted only when no other process is accessing the file. A resource that does not require exclusive access can be concurrently accessed by multiple processes.

**Invariant Behavior of the Shared Resource** Suppose a synchronizer were controlling accesses to a bounded stack. Correct use of the stack demands that the stack neither overflow nor underflow. Thus, accesses to the stack should be permitted only if this invariant property is not violated by the access.

**Sequencing Accesses to the Resource** Some resource accesses need to be permitted according to some strict sequence. Consider a one slot buffer, with operations "put" for leaving a message in the buffer and "get" for retrieving it. For correct use of the buffer, once a process puts a message, another "put" can be performed only after a "get" is executed by some process.