Correctness of Programs and Protocols through Randomization

(Extended Abstract)

Michael O. Rabin
Harvard University and the Hebrew University of Jerusalem

The problem of proving the correctness of programs and protocols is long standing and of central importance. Numerous formal systems and logics were developed and are being used to tackle this difficult task. When it comes to involved protocols, the formal proofs of correctness, where possible, are hard and often require an understanding of the behavior of the protocol so deep as to amount to an informal proof of correctness. At the same time, there are instances of practically significant protocols which were for a long time accepted as being correct, until an attempt at a formal proof of correctness has actually uncovered an error. As is well known from practice, the difficulty of establishing the correctness of protocols usually arises out of the presence of concurrent actions and the complexity of attempting to follow the myriad of ways in which these actions may interleave to produce unintended behavior. In the case of security protocols, proving correctness involves delineation of all possible ways in which an adversary or corrupt participants may subvert the protocol.

We propose an alternate approach employing randomization. It is well known that in many cases, randomized algorithms and protocols for the solution of specific problems are considerably simpler as well as more efficient than the “classical” deterministic solutions. Also, for some important problems, one can prove that no deterministic solution exist. It is the relative simplicity of the randomized solutions that we propose to utilize for facilitating the task of proving their correctness. In methodological terms, what we do is to exchange the need to account for all possible behaviors of a protocol, with the task of proving that certain random variables (such as coin-tosses) employed in the randomized solution are independent. While the latter task may be subtle, it involves well known methods of probability theory and in most cases is not challenging even when dealing with problems that in the deterministic setting are very hard.

We shall demonstrate the efficacy of the randomized approach by dealing with two classical problems which are very important in their own right and which were the subject of hundreds of papers.

The first one is the Critical Section Problem. Processes $P_1, \ldots, P_n$ are computing concurrently. From time to time any one of the processes, say $P_1$, may need to enter a critical section $C$ (say use a shared resource). It is required that while $P_1$ is in $C$, no other process will enter the critical section. This requirement
is defined as *mutual-exclusion*. A proper solution for this problem should also be deadlock free, i.e. that the attempt to insure mutual-exclusion will not result in a situation that none of of the processes wishing to enter the critical section can do so. In addition, it may be desirable that the solution impose some fair distribution of the accesses to \( C \) amongst the processes.

Clearly the processes \( P_1, \ldots, P_n \) must communicate amongst themselves so as to avoid cohabitation of the critical section and to impose fair sharing. There are many models for the mode of communications that were proposed, and for each model several protocols were proposed and sometimes proved correct. We shall consider an important model suggested by L. Lamport. With every process \( P_i \) we associate a variable \( R[i] \), owned by \( P_i \). This means that only \( P_i \) can update (write) \( R[i] \), but every other \( P_j \) can read this variable. There is a natural justification for considering such an arrangement. Using these variables, Lamport gave a number of protocols achieving deadlock free and fair mutual exclusion. Lamport and many others gave formal correctness proofs for such protocols. The difficulty of this problem is highlighted by the fact that in some instances, errors were discovered despite correctness proofs.

A great deal of the challenge in proving correctness of protocols formulated in Lamport’s model arises from the stringent semantics he imposes on reads and writes of the variables \( R[i] \). Namely, Lamport stipulates that a write or a read operation on \( R[i] \) is not instantaneous but extends in time. Furthermore, if some \( P_j \) reads \( R[i] \) while \( P_i \) updates that variable, the value returned to \( P_j \) may be arbitrary. Again, there are good reasons for such a stipulation.

An examination of the protocols for the Critical Section Problem in Lamport’s model reveals that, absent the possible interference between write and read operations on the same owned variables, verification is quite straightforward. Thus we finesse the obstacle posed by the stringent write/read semantics, by introducing a new construct of a *faithful version* of an owned variable \( R[i] \). The methodology is completely general and applies to any protocol or concurrent program written in Lamport’s model.

Let \( R[i] \) be a variable owned by \( P_i \). We define a data structure \( F(R[i]) \) and two procedures \( \text{Write}_i(F(R[i])) \) and \( \text{Read}(F(R[i])) \). The Write and Read procedures are randomized. The faithful version of \( R[i] \) has the following strong properties.

If a process \( P_j \) executes \( \text{Read}(F(R[i])) \) and this execution does not overlap in actual time an execution of \( \text{Write}_i(F(R[i])) \) by \( P_i \), then the Read will return to \( P_j \) the value of \( R[i] \) written by \( P_i \) in the latest execution of \( \text{Write}_i(F(R[i])) \) preceding the start of \( \text{Read}(F(R[i])) \) by \( P_j \).

If \( P_j \)'s execution of \( \text{Read}(F(R[i])) \) does overlap an execution of \( \text{Write}_i(F(R[i])) \) by \( P_i \), then the Read will either return to \( P_j \) a value of \( R[i] \) written by \( P_i \) between the start and end of the execution of \( \text{Read}(F(R[i])) \), or return the value “undefined”.

In summation: \( \text{Read}(F(R[i])) \) will always return a correct value for \( R[i] \), or else signal to \( P_j \) that an interference has occurred. Note that this is achieved under the stringent condition that for the memory cells comprising the faithful version \( F(R[i]) \) of \( R[i] \), which are all owned by \( P_i \), if a read by \( P_j \) overlaps in