Wakeup under Read/Write Atomicity

Prasad Jayanti      Sam Toueg
Department of Computer Science
Upson Hall, Cornell University
Ithaca, New York 14853
e-mail(prasad@cs.cornell.edu, sam@cs.cornell.edu)

Abstract

We study the wakeup problem in an asynchronous system of anonymous processes that are identically coded. Informally, a protocol solves wakeup if, in all infinite fair runs of the protocol, every process eventually learns that every process has executed at least one step and is therefore ‘awake’ [FMRT90]. We study this problem in the context of two models of shared memory communication. In the first model, each word of shared memory may be read and written by every process. In the second model, each process has a register which only that process may write, but all processes may read. We show that wakeup is not solvable in the first model. For the second model, we derive matching lower and upper bounds on the amount of shared memory necessary to solve wakeup. We also study the feasibility of leader election and consensus in the above two models.

1 Introduction

Self-stabilization refers to the ability of a distributed system to eventually start behaving correctly even if the initial states of the processes and the shared memory are arbitrary. Following the pioneering work of Dijkstra [Dij74], there has been a lot of research in this area. Although self-stabilization adds fault tolerance to non-terminating protocols such as mutual exclusion, it is not appropriate, as noted by Fischer et al [FMRT90], for terminating protocols such as wakeup, consensus, and leader election. To see this, note that there will be no system execution if the initial state of each process is a termination state. Fischer et al have, therefore, proposed an alternate model of interest to terminating protocols [FMRT90]. In their model, the system consists of identically programmed anonymous asynchronous processes communicating through shared memory. Unlike self-stabilizing systems, the program counter of each process points to the appropriate instruction initially. The initial state of shared memory, however, is arbitrary as in self-stabilizing systems. This gives a process the flexibility to start its participation
in the protocol at any time it chooses without any global synchronization\(^1\).

In this paper, we study primarily the \textit{wakeup} problem in the model outlined above. Leader election and consensus are the other problems studied. Informally, a wakeup protocol is one in which, in every infinite fair run, each process eventually learns that every other process has woken up and begun participating in the protocol. The non-triviality condition is the obvious one: process \(p\) cannot learn that process \(q\) has woken up unless \(q\) has taken at least one step of the protocol.

The \textit{wakeup} problem was first defined and studied in [FMRT90]. In the communication model of [FMRT90], there is a single shared variable in the entire system and processes use this shared variable for communication. It is assumed that each process may do a \textit{test-and-set} operation on this variable, and that this operation is atomic. Our first result shows that, if in the above model one assumes only read/write atomicity, even a weaker version of the wakeup problem becomes unsolvable.

We then consider the \textit{wakeup} problem in an alternate model of communication: we assume that there is a communication register associated with each process and that this register may be written only by that process, but may be read by all processes. This is a Single-Writer-Multi-Reader (SWMR) model in contrast to the earlier described Multi-Writer-Multi-Reader (MWMR) model. We present a terminating \textit{wakeup} protocol when each communication register is a three-valued regular register\(^2\). Following that, we prove the optimality of this protocol by showing that \textit{wakeup} is not solvable if each communication register is only a boolean.

In this paper, we also consider two other fundamental problems: leader election and consensus (see [Ita90] and [CIL87] for further references). We show that leader election is not solvable in both SWMR and MWMR models. Consensus, on the other hand, is solvable in SWMR but not in MWMR. Impossibility of consensus in MWMR is not due to process failures since they are not allowed in our model. It is interesting to contrast this result with [CIL87] in which consensus is proved impossible in the presence of failures, even if the model is not symmetric.

The paper is organized as follows. In section 2, we describe the model and specify the \textit{wakeup} problem. In section 3, we prove that even the weakest form of \textit{wakeup} is not solvable in the MWMR model. In section 4, we focus on the SWMR model. First, we present a terminating \textit{wakeup} protocol when each communication register is a three-valued regular register. Then we show that \textit{wakeup} is not solvable when each communication register is only a boolean. We present results on consensus and leader election in section 5. We conclude in section 6.

\(^1\)This is in contrast to most models in which each process \(p\) participating in the protocol must synchronize with the process \(q\) responsible for the initialization of shared memory such that this initialization by \(q\) precedes the steps of \(p\).

\(^2\)A register \(r\) is \textit{regular} if the value returned by every read operation is either the value written into \(r\) by the most recent write operation that completed before that read operation started or the value written into \(r\) by some write operation overlapping with that read operation.