Asynchrony from Synchrony

Yehuda Afek\(^1\) and Eli Gafni\(^2\)

\(^1\) Blavatnik School of Computer Science, Tel Aviv University
\(^2\) Computer Science Department, University of California, Los Angeles

Abstract. A synchronous message passing complete network with an adversary that may purge messages is used to precisely model tasks that are read-write wait-free computable.

In the past, adversaries that reduce the computational power of a system as they purge messages were studied in the context of their ability to foil consensus. This paper considers the other extreme. It characterizes the limits on the power of message-adversary so that it cannot foil the solution of tasks which are read-write wait-free solvable but can foil the solution of any task that is not read-write wait-free solvable. Put another way, we study the weakest message-adversary which allows for solving any task that is solvable wait-free in the read-write model.

A remarkable side-benefit of this characterization is a simple, as simple as can be, derivation of the Herlihy-Shavit condition that equates the wait-free read-write model with a subdivided-simplex. We show how each step in the computation inductively takes a subdivided-simplex and further subdivides it in the simplest way possible, making the characterization of read-write wait-free widely accessible.

Keywords: shared memory, distributed algorithms, wait-free, subdivided simplex, asynchronous computability.

1 Introduction

The seminal FLP result \cite{15} shows that in a message-passing system with the possibility of a single processor failing by stopping (1-resilient system) consensus is not solvable. From a programmer’s point of view this allows her to instruct processors to wait to receive messages from all processors but one (we assume the number of processors, \(n\) satisfies \(n > 2\)). That is, she may write her program to progress in rounds where in each round a processor sends a message to all and then waits to receive messages sent at this round from all but one processor. Thus she can view the model of computation as a synchronous system with a message-adversary that is allowed to purge at most one message incoming to each processor in a round. From FLP it follows that such a message-adversary can foil consensus. It is easy to see that if we change the message-adversary so it can purge any but at most \(n - 2\) messages in a round then consensus could be reached.

Thus, we are led to consider a message-passing systems on \(n\) processors that progress in rounds. In each round all processors send to all. There is a synchronous indication of an end of a round. If \(p_i\) did not receive a message from \(p_j\)
by the end of the round it has to be attributed to an adversary that purged the
message. We consider adversaries as a constraint on the message pattern they
may purge. This constraint applies uniformly to all rounds, and the patterns the
adversary may purge in a round is independent of what it actually purged in
previous rounds.

Clearly the weaker the message-adversary, the more tasks the programmer
can solve, and vice versa, the more powerful adversary, the more it can purge,
the less tasks the programmer can solve.

In this paper we introduce a general definition of message-adversaries and
investigate the relations between the adversaries power and the tasks that may
be computed in their presence. Specifically we identify the weakest message-
adversary with which any task that is wait-free solvable in a read-write shared
memory system is solvable in a synchronous network governed by that message-
adversary. To recap, we consider a complete synchronous message passing di-
rected network employing a full information algorithm by which each processor
in each round sends all its history to all, and by the end of the round collects
all the messages that have arrived on its links, i.e., those that were not purged
by the adversary. In each round the adversary may remove a subset of the mes-
sages that have been sent. The subset removed in one round is independent of
the subsets removed in previous rounds. Furthermore, the removal of a message
might be asymmetric, a message removed on a link in one direction does not
imply the removal of the message in the other direction.

To specify an adversary $AD$ in a round we view a successful message from
processor $p_i$ to processor $p_j$ as a directed edge from node $i$ to node $j$. An ad-
versary $AD$ is a set of directed graphs such that in each round there is a digraph
$G$, $G \in AD$ such that the message sent on each link in $G$, successfully reaches
the other side. Notice, the adversary is restricted not to purge more than it
is allowed, but can always leave more successful messages, i.e., the successful
messages may induce a graph $H$ such that $\exists G, G \in AD$ which is a subgraph
of $H$.

For instance, the message adversary AD-1-res(incoming), corresponding to 1-
resilient systems is specified by all the directed graphs on $n$ nodes in which the
in-degree of each node is $n - 2$. Thus it takes $(n - 1)^n$ graphs to specify this
adversary.

An adversary $AD$ characterizes the asynchronous shared-memory model $M$,
if all the tasks it allows to solve are exactly all the tasks solvable by $M$. In
this paper we restrict our attention to message-adversaries that characterize
the class of tasks are wait-free solvable in an asynchronous read-write shared-
memory (RWWF) model. We show that the adversary that captures RWWF
is the Traversal Path, TP adversary. The TP adversary can remove any set of
messages in a round as long as the directed graph induced by the messages it
leaves behind is a not necessarily simple, path of messages that goes through
all the nodes (i.e., TP contains all possible paths that satisfy the above). Notice
that TP satisfies the property that it leaves messages such that for any pair $p_i$,