Verification of a Distributed Summation Algorithm

Frits Vaandrager

Department of Software Technology, CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
fritsv@cwi.nl

Abstract. A correctness proof of a variant of Segall's Propagation of Information with Feedback protocol is outlined. The proof, which is carried out within the I/O automata model of Lynch and Tuttle, is standard except for the use of a prophecy variable. The aim of this paper is to show that, unlike what has been suggested in the literature, assertional methods based on invariant reasoning support an intuitive way to think about and understand this algorithm.

1 Introduction

Reasoning about distributed algorithms appears to be intrinsically difficult and will probably always require a great deal of ingenuity. Nevertheless, research on formal verification has provided a whole range of well-established concepts and techniques that may help us to tackle problems in this area. It seems that by now the basic principles for reasoning about distributed algorithms have been discovered and that the main issue that remains is the problem of scale: we know how to analyze small algorithms but are still lacking methods and tools to manage the complexity of the bigger ones.

Not everybody agrees with this view, however, and frequently one can hear claims that existing approaches cannot deal in a 'natural' way with certain types of distributed algorithms. A new approach is then proposed to address this problem. A recent example of this is a paper by Chou [4], who offers a rather pessimistic view on the state-of-the-art in formal verification:

At present, reasoning about distributed algorithms is still an ad hoc, trial-and-error process that needs a great deal of ingenuity. What is lacking is a practical method that supports, on the one hand, an intuitive way to think about and understand distributed algorithms and, on the other hand, a formal technique for reasoning about distributed algorithms using that intuitive understanding.

In his paper, Chou proposes an extension of the assertional methods of [2, 7, 8, 9, 10, 12, 15], and argues that this extension allows for a more direct formalization

* The author was supported by Esprit BRA 7166 COGNUT 2. This paper is dedicated to P.C. Baayen who, like the root node in Segall's algorithm, initiated a cascade of mathematical activity.
of intuitive, operational reasoning about distributed algorithms. To illustrate his method, Chou discusses a variant of Segall's PIF (Propagation of Information with Feedback) protocol [19]. A complex and messy proof of this algorithm using existing methods is contrasted with a slightly simpler but definitely more structured proof based on the new method.

Is the process of using assertional methods based on invariant reasoning ad hoc? Personally, I believe it is not. On the contrary, I find that these methods provide significant guidance and structure to verifications. After one has described both the algorithm and its specification as abstract programs, it is usually not so difficult to come up with a first guess of a simulation relation from the state space of the algorithm to the state space of the specification. In order to state this simulation, it is sometimes necessary to add auxiliary history and prophecy variables to the low-level program. By just starting to prove that the guessed simulation relation is indeed a simulation, i.e., that for each execution of the low-level program there exists a corresponding execution of the high-level program, one discovers the need for certain invariants, properties that are valid for all reachable states of the programs. To state these invariant properties it is sometimes convenient or even necessary to introduce auxiliary state variables. Frequently one also has to prove other auxiliary invariants first. The existence of a simulation relation guarantees that the algorithm is safe with respect to the specification: all the finite behaviors of the algorithm are allowed by the specification. The concepts of invariants, history and prophecy variables, and simulation relations are so powerful that in most cases they allow one to formalize the intuitive reasoning about safety properties of distributed algorithms. When a simulation (and thereby safety) has been established, this simulation often provides guidance in the subsequent proof that the algorithm satisfies the required liveness properties: typically one proves that the simulation relates each fair execution of the low-level program to a fair execution of the high-level program. Here modalities from temporal logic such as "eventually" and "leads to" often make it quite easy to formalize intuitions about the liveness properties of the algorithm.

As an illustration of the use of existing assertional methods, I outline in this paper a verification within the I/O automata model [12, 13] of the algorithm discussed by Chou [4]. Altogether, it took me about two hours to come up with a sketch of the proof (during a train ride from Leiden to Eindhoven), and about three weeks to work it out, polish it, and write this paper. The proof is routine, except for a few nice invariants and the use of a prophecy variable. Unlike history variables, which date back to the sixties [11], prophecy variables have been introduced only recently [1], and there are not that many examples of their use. My proof is not particularly short, but it does formalize in a direct way my own intuitions about the behavior of this algorithm. It might very well be the case that for more complex distributed algorithms new methods, such as the one of Chou [4], will pay off and lead to shorter proofs that are closer to intuition. This paper shows that invariant based assertional methods still work very well for a variant of Segall's PIF protocol.

The structure of this paper is as follows. Section 2 describes the algorithm formally as an I/O automaton. Section 3 presents the correctness criterion and the proof that the algorithm meets this criterion. Finally, Section 4 contains some concluding remarks. For reasons of space all the proofs of the lemmas and theorems have been omitted in this extended abstract. They can be found in the full version [21],