Transformational Verification of Parameterized Protocols Using Array Formulas

Alberto Pettorossi\textsuperscript{1}, Maurizio Proietti\textsuperscript{2}, and Valerio Senni\textsuperscript{1}

\textsuperscript{1} DISP, University of Roma Tor Vergata, Via del Politecnico 1, I-00133 Roma, Italy
pettorossi@info.uniroma2.it, senni@disp.uniroma2.it
\textsuperscript{2} IASI-CNR, Viale Manzoni 30, I-00185 Roma, Italy
proietti@iasi.rm.cnr.it

Abstract. We propose a method for the specification and the automated verification of temporal properties of parameterized protocols. Our method is based on logic programming and program transformation. We specify the properties of parameterized protocols by using an extension of stratified logic programs. This extension allows premises of clauses to contain first order formulas over arrays of parameterized length. A property of a given protocol is proved by applying suitable unfold/fold transformations to the specification of that protocol. We demonstrate our method by proving that the parameterized Peterson’s protocol among $N$ processes, for any $N \geq 2$, ensures the mutual exclusion property.

1 Introduction

Protocols are rules that govern the interactions among concurrent processes. In order to guarantee that these interactions enjoy some desirable properties, many sophisticated protocols have been designed and proposed in the literature. These protocols are, in general, difficult to verify because of their complexity and ingenuity. This difficulty has motivated the development of methods for the formal specification and the automated verification of properties of protocols. One of the most successful methods is model checking \cite{5}. It can be applied to any protocol that can be formalized as a finite state system, that is, a finite set of transitions over a finite set of states.

Usually, the number of interacting concurrent processes is not known in advance. Thus, people have designed protocols that can work properly for any number of interacting processes. These protocols are said to be parameterized with respect to the number of processes. Several extensions of the model checking technique based upon abstraction and induction have been proposed in the literature for the verification of parameterized protocols (see, for instance, \cite{3,18,27,29}). However, since the general problem of verifying temporal properties of parameterized protocols is undecidable \cite{2}, these extensions cannot be fully mechanical.

In this paper we propose an alternative verification method based on program transformation \cite{4}. Our main objective is to establish a correspondence between protocol verification and program transformation, so that the large number of semi-automatic techniques developed in the field of program transformation can be applied to the verification of properties of parameterized protocols.
Since arrays are often used in the design of parameterized protocols, we will consider a specification language that allows us to write array formulas, that is, first order formulas over arrays. We will specify a parameterized protocol and a property of interest by means of a logic program whose clause bodies may contain array formulas. Our verification method works by transforming this logic program, in which we assume that the head of the clause specifying the property has predicate \( prop \), into a new logic program where the clause \( prop \leftarrow \) occurs. Our verification method is an extension of many other techniques based on logic programming which have been proposed in the literature \[7, 9, 11, 19, 22, 23\].

We will demonstrate our method by considering the parameterized Peterson’s protocol \[20\]. This protocol ensures mutually exclusive use of a given resource which is shared among \( N \) processes. The number \( N \) is the parameter of the parameterized protocol. In order to formally show that Peterson’s protocol ensures mutual exclusion, we cannot use the model checking technique directly. Indeed, since the parameter \( N \) is unbounded, the parameterized Peterson’s protocol, as it stands, cannot be viewed as a finite state system. Now, one can reduce it to a finite state system, thereby enabling the application of model checking, by using the above mentioned techniques based on abstraction \[3\]. However, it is not easy to find a powerful abstraction function which works for the many protocols and concurrent systems one encounters in practice.

In contrast, our verification method based on program transformation does not rely on an abstraction function which is applied once at the beginning of the verification process, but it relies, instead, on a generalization strategy which is applied on demand during the construction of the proof, possibly many times, depending on the structure of the portion of proof constructed so far. This technique provides a more flexible approach to the problem of proving properties of protocols with an infinite state space.

The paper is structured as follows. In Section 2 we recall the parameterized Peterson’s protocol for mutual exclusion which will be used throughout the paper as a working example. In Section 3 we present our specification method which makes use of an extension of stratified logic programs where bodies of clauses may contain first order formulas over arrays of parameterized length. We consider properties of parameterized protocols that can be expressed by using formulas of the branching time temporal logic CTL \[5\] and we show how these properties can be encoded by stratified logic programs with array formulas. Then, in Section 4, we show how CTL properties can be proved by applying unfold/fold transformation rules to a given specification. In Section 5 we discuss some issues regarding the automation of our transformation method. Finally, in Section 6 we briefly discuss the related work in the area of the verification of parameterized protocols.

## 2 Peterson’s Mutual Exclusion Protocol

In this section we provide a detailed description of the parameterized Peterson’s protocol \[20\]. The goal of this protocol is to ensure the mutually exclusive access