Axiomatic Treatment of Processes with Shared Variables Revisited

Olaf Owe
Department of Informatics, University of Oslo, Norway

Dedicated to the memory of Jan Helge Dæhlin, 1959-1989

Keywords: Specification; Verification; Concurrency; Shared variables

Abstract. The aim of this paper is to develop simple and practically useful formalisms for reasoning about processes with shared variables. Our approach is based on the axiomatic system described by Neelam Soundararajan. In contrast to that work, our formalism is first derived from a model; this guarantees soundness and completeness of the formal proof system, with respect to the model. As an additional advantage the rules become simpler than those of Soundararajan; in particular, the local assertions may freely refer to shared variables; and we remove the explicit use of the compatibility predicate.

Next we augment the formalism by allowing global invariants, which may refer to shared variables (including shared histories), but with a different semantics than in the local assertions. The augmented system makes reasoning simpler in the sense that reasoning about the past is replaced by reasoning about the present. Finally we suggest a sufficiently complete set of mythical (auxiliary) variables free from embedded program structure. We demonstrate our formalism on some examples.

1. Introduction

We focus on formalisms for specification and reasoning about parallelism involving shared variables. Many such formalisms have been suggested [Ash75, OwG75, Dah79, Lam80, OwL82, Jon83, Sou84, MaP84, Dæh87, Lam88, Sti88, GjM89, Stø90]. In particular we will build on the work of Neelam Soundararajan [Sou84], which is based on histories (time sequences), in the tradition of [Kah74, Dah79].

Correspondence and offprint requests to: O. Owe, Department of Informatics, University of Oslo, Norway.
This formalism is elegant, in the sense that it is sound and (relatively) complete without need of proofs of freedom of interference, and without use of mythical (auxiliary) variables corresponding to the program counter. And it is compositional in the sense that the processes can be specified and verified separately. However in practice, much of the program structure must be embedded in the mythical history variables, and often specifications can be quite unreadable, and reasoning, even about simple facts, is difficult. Our goal is to develop a formalism which makes reasoning simple. We believe that an important step towards practically useful formalisms is to find ways to reduce the need of program structure embedded in mythical variables.

A major difficulty when reasoning about parallelism involving shared variables is that the normal Hoare rules [Hoa69] are not sound if the assertions local to a process \( i \) refer to shared variables. And if the assertions only refer to local variables, little can be proved about the parallel composition by the `cobegin-coend' statement. As shown in [Sou84], this problem can be avoided by letting the local assertions of process \( i \) refer to a mythical program variable \( h_i \), which is the history of events involving access to shared variables as seen from process \( i \); provided the mythical effects on \( h_i \) are catered for. (We will specify these mythical assignments below.) By the principle of compatibility of the local histories [SoD82], one may, at the `coend' state, reason about the total history of events as well as the final values of the shared variables.

In Section 2, we define the semantics, in the sense of partial correctness, of a concurrent program as a non-deterministic sequential program, based on the mythical program variables \( h_i \) and the global history \( h \). Since this model is sequential we may reason about it by usual Hoare logic; and we may use the model to derive Hoare-like proof rules for concurrent programs; this is done in Section 3. The derived Hoare rules have several advantages compared with those in [Sou84], which were only partially motivated by a model. In the present formalism, local assertions may freely refer to the shared variables. Examples show that this freedom is practically useful. Furthermore, the rule for parallel composition is expressed without the compatibility predicate, which is used by several authors [SoD82, Sou84, Dæh87, GjM89]. In this respect our formulation is closer to the work of Sigurd Meldal [Mel86, Mel89]; however, that work does not consider shared variables.

Due to the expressiveness of the history concept, we find our model intuitively correct; and omit a further discussion of the correctness of the model. By soundness of Hoare logic, it follows that the derived rules are sound with respect to the model. By (relative) completeness of Hoare logic, it follows that the derived rules are (relatively) complete, with respect to our model, because the derived rules are optimally strong.

In Section 4, we add rules for handling global (shared) invariants. A global invariant is common to all processes and may refer to shared variables and the global history. Such an invariant may serve as a helpful design invariant in top-down program development, capturing essential ideas of a parallel composition under construction. In our formalism, both local assertions and the global invariant may freely refer to shared variables, but with different semantics, reflecting the different locations of view. In other respects our formalism is quite similar to that of [Dah89]. We show how to integrate reasoning about invariants with local reasoning such that both kinds of reasoning benefit from each other. This allows much of the reasoning normally done at the `coend' state with the system in [Sou84], to be decomposed into several verification conditions. Thereby,