0. Abstract.

We believe that axiomatic reasoning about program behaviour should be based directly on a semantic model specifically tailored for that purpose. Moreover, the structure of the semantic model should be used directly to suggest the structure of an assertion language for expressing program properties. It is desirable, therefore, to adopt a semantics with as clean and simple a structure as possible, so that one can use assertions with simple syntactic structure and build a clean and simple axiomatic proof system for program properties. By basing the proof system closely on the underlying semantic model, one is able to use the semantic model directly in establishing the soundness and relative completeness of the proof system; these tasks are made less difficult if the semantics has a simple structure. We illustrate these ideas by applying them to a small programming language, a simple block-structured imperative language which allows sharing or aliasing among identifiers. Although the language is rather simple and is certainly far from being a fully fledged programming language, it exhibits enough semantic features to merit a detailed semantic investigation and serves well to illustrate our methodology for designing an axiomatization. We first define a standard semantics and discuss the full abstraction problem for this language. We define a semantic relation called sharing equivalence, and show that the standard semantics is fully abstract "up to sharing equivalence". We then construct a semantics based directly on sharing equivalence classes, which is fully abstract. We establish some important semantic properties, and use them in designing an axiomatic proof system for partial correctness properties of programs.