
Type of paper: This paper is an enhanced method paper: it shows a way to simplify proofs of program correctness by mapping between abstract and concrete representations of data types.

Correlation to the real-world setting: The work is motivated by the understanding that programmers will design abstract algorithms and data types first, and then progress to concrete algorithms and data types.

What is the hypothesis: This work proposes building a proof that the concrete data type correlates to the abstract data type. Doing this allows any proof of the correctness of the abstract algorithm to be mechanically transformed into a proof of correctness of the concrete algorithm.

Validation: Show a toy example of a class that implements a set. Quickly sketch the proofs needed to prove the correctness of a concrete implementation of the class, and outline the technique to transform a proof of the abstract algorithm into a proof of the concrete algorithm.

Result: Some of the insights are rather interesting: “benevolent side-effects” are responsible for most of the casts that I had to implement in my C++ code. There are two interesting overlaps with Parnas’s work. First, the author seems to be advocating a form of information hiding in the form of scope rules. Second, the author realizes that providing greater visibility into the classes will increase the number of invariants which the code must observe. His observation that the actual types involved could be parameterized without changing the proofs seems to be a good basis for generics. His observation that copy by value is simpler to reason about than copy by address is a good insight.

Maturity of the work: The work is immature; there is no argument that it can be scaled up and applied to a real program.

Do you believe the result, and why? Based on the author’s examples, I believe his assertions that certain ways of implementing the concrete classes will complicate the proofs. However, without prior experience I would not be convinced to avoid these implementations based on the evidence in this paper. Too many things (like the limit of 100 elements in the set) escape into the invariants for me to believe that the approach will scale well. In fact, won’t these new invariants complicate the translation of the proofs? And, of course, there are the classic problems of finding a workable definition of correctness for the abstract program and proving that it conforms to this definition.