

Multidimensional Program Verification

Ed Ashcroft

Computer Science and Engineering Department

Arizona State University

Tempe AZ 85287

ed.ashcroft@asu.edu

phone: +1 602 9647544 fax: +1 602 965 2751

April 13, 1995

Abstract

The language `Lucid` was originally invented as a language for which proofs of program properties are relatively straightforward. Proofs are mainly based upon conventional mathematical reasoning, together with some basic properties of the `Lucid` operations and some new proof techniques, such as `Lucid Induction`.

Now that `Lucid` has been extended to deal with multidimensional objects, it is natural to wonder whether new basic properties and new proof techniques are necessary and, even, available. What has been discovered is that a property of multidimensional objects, a property that seems to be unnecessary for the formal mathematical semantic definition of multidimensional `Lucid`, namely *rank*, is very useful in proving properties of programs and, we claim, in gaining intuitive understanding of the meanings of programs. (The concept of rank is also useful in the compilation of `Lucid` and `GLU` programs.)

Multidimensional objects were introduced in Section 3 of Chapter VII (*Beyond Lucid*), of the first `Lucid` book. There, objects had an infinite number of dimensions but dimensions were never referred to or acted upon specifically, except for the first one. (The operations `initial`, `rest`, and `cby` all worked on the first "space" dimension.) The language worked and was useful but experience using it showed that it could be clumsy and constraining. This has led to the multidimensionality features of the language described in the new book.

In *New Lucid*, dimensions are named and can be worked upon individually. New, local dimensions can be set up, and, in fact, are used for subcomputations, thereby

replacing the `is current` feature of Old Lucid. The new dimensions are declared in where clauses, just as frozen (`is currented`) variables were in Old Lucid. The big difference is that in Old Lucid frozen variables had to be declared, while other variables were naturally fresh, whereas in New Lucid variables are naturally frozen when new dimensions are introduced, while, to be made fresh, variables have to be explicitly reactivated by applying the function `realign`, to lay them out in the new dimension.

The Old Lucid operations, such as `first`, `next`, and `fby`, can be used for any dimension. All we have to do is *focus* them onto the dimension in question, say dimension `d`, by writing `first.d`, `next.d`, and `fby.d`. We extend this idea to functions, so that we can focus functions onto whatever dimensions we wish. To be able to do this, the function definition must indicate what are the *formal dimensions* that are going to be focussed onto *actual dimensions* during calls of the function. We say that such functions are *dimensionally abstracted*. (Since variables are nullary functions, they, too, can be dimensionally abstracted.) In the program verification context, this immediately raises the question -- can *properties* proved of dimensionally abstracted functions and variables be focussed upon arbitrary dimensions?

One of the most interesting techniques used in Lucid proofs is Lucid Induction. Originally, Lucid Induction worked on a single dimension, the "time" dimension. Now that there are several dimensions, there are several different versions of Lucid Induction, not just working on the separate dimensions but also working on several dimensions together.

Lucid Induction on a single dimension `a` can be stated as follows: If we can prove *first.a P* and we can prove $P \rightarrow \text{next.a } P$ then we can infer P . (\rightarrow is pointwise implication: if $A \rightarrow B$ then, for all contexts, if A is true in that context then B is true in the same context.) In order to get a pointwise implication, we must employ *pointwise reasoning*. (This was all explained and formalized in a paper by Ashcroft and Wadge in SIAM Journal of Computing, back in 1976.)

Single-dimension Lucid Induction can be used to prove many things about multi-dimensional objects, but some things require multidimensional Lucid Induction. For example, the two-dimensional Lucid Induction Rule on two dimensions `a` and `b` can be stated as follows: If we can prove *first.a P* and we can prove *first.b P* and we can prove $P \rightarrow \text{next.a next.b } P$ then we can infer P . (Again, \rightarrow is pointwise implication: if $A \rightarrow B$ then, for all contexts, if A is true in that context then B is true in the same context.) In order to get a pointwise implication, we must employ *pointwise reasoning*. The generalization of the formalization of pointwise reasoning introduced in the 1976 paper is quite straightforward.

In this paper we give the details of proofs promised in the new book.