Deriving Algorithms

A derivation is a proof of correctness written backwards. We usually develop an algorithm to satisfy some properties. One could therefore say that we informally derive the algorithm from an informal proof that it satisfies those properties. A formal derivation is one that obtains the algorithm from its correctness properties by formal manipulations that guarantee the resulting algorithm to be correct.

I have heard of just one case of an interesting algorithm being discovered by this kind of formal derivation. I've always discovered interesting algorithms by a mysterious combination of informal reasoning and intuition. Only after coming up with the algorithm—and, since the development of TLC, only after model checking it—did I try to prove it rigorously.

We can formally derive the two-phase handshake protocol from the *Alternation* algorithm as follows. We first define *InitPC* and *NextPC* to be the formulas obtained by substituting $p \oplus c$ for b in formulas *Init* and *Next* of module *Alternation*. We then strengthen these formulas (find formulas that imply them) to obtain formulas *Init*2*ph* and *Next*2*ph* that have the right form to be the translation of a PlusCal algorithm. Our construction of these formulas guarantees that a module with the behavior specification defined by *Init*2*ph* and *Next*2*ph* (or by the PlusCal algorithm whose translation produces them) implements *Alternation* under the refinment mapping $\overline{b} \triangleq p \oplus c$, $\overline{box} \triangleq box$.

CLOSE