To prove that a program satisfies its specification, it is more intellectually manageable to construct the program such that its structure enables the programmer to “give convincing argument for its correctness” [2], or even serves as a proof by itself, making the program manifestly correct.

### Solving the Dutch National Flag problem via datatype ornamentation

**Josh Ko (University of Oxford)**

In modern dependently typed languages like Agda [4], data can be specified such that they satisfy certain properties by construction. Consequently, programs constructing those data are manifestly correct without need for separate proofs, since being able to construct the data implies that the properties are indeed established.

What is algebraic ornamentation?

An algebraic ornamentation adds an extra index to a datatype such that the index in the type of an element is always the value computed by a particular fold on that element. This technique was identified and for -

by algebraic ornamentation, the length of the unknown section is integrated into the type of the Dutch vectors. This simplifies the type of firstUnknownColour, and the length will serve as an explicit termination measure. We then attempt to describe how to reduce the length of the unknown section by one, which requires a case analysis on the result of firstUnknownColour. The case analysis does not directly reveal more information about the input Dutch vector, however — to do so, that result has to be exposed in the type by another algebraic ornamentation.

What properties to encode in datatypes are often discovered only gradually during development. Ornamentation suggests a way of supporting incremental specification of precise datatypes to match our development patterns.