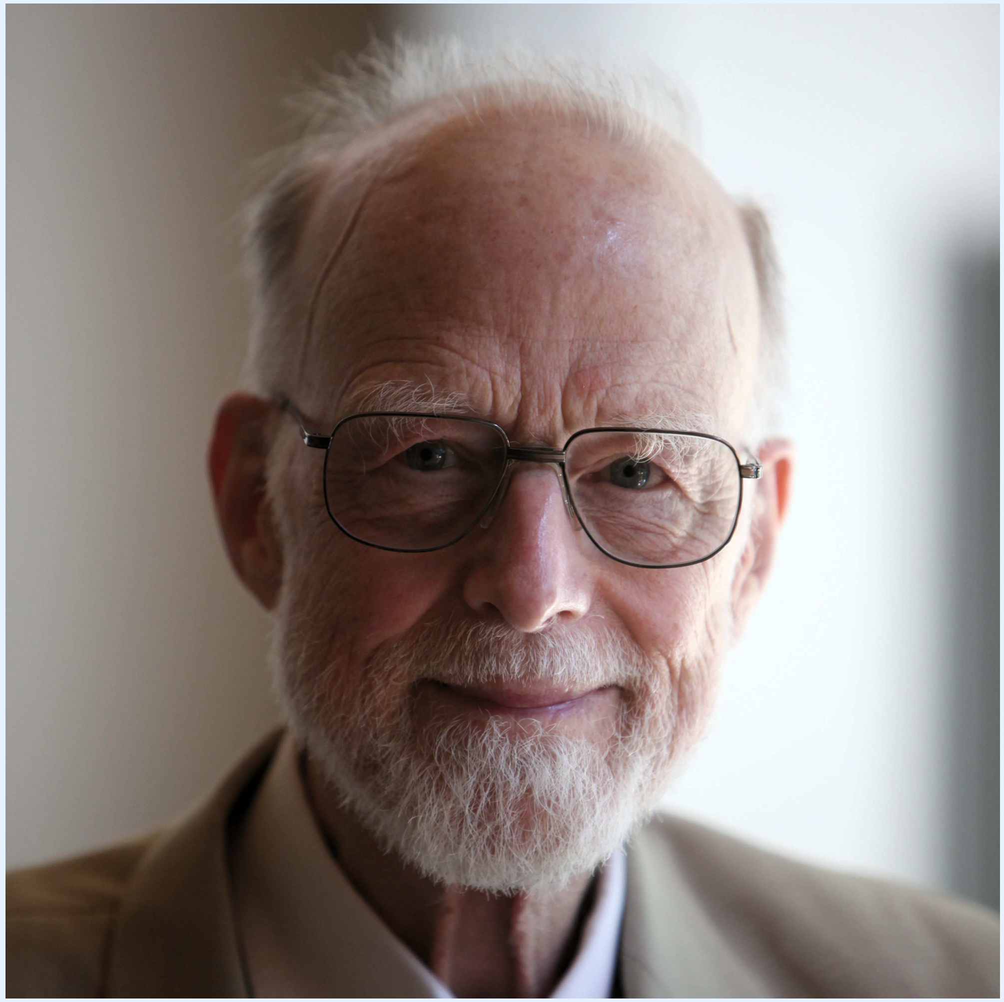
An Axiomatic Basis for Computer Programming

Christopher Mark Gore

cgore.com

Thursday, March 5, AD 2015

Sir Charles Antony Richard Hoare FRS FREng



You can call me Tony!

He’s Really Smart!

- Quick Sort (a.k.a. the only good sort),
- Quick Select (a.k.a. the only good select),
- Hoare Logic (this paper is its beginnings),
- CSP (Communicating Sequential Processes): formalized message passing,
- Null references *(or is that actually a bad thing?)*

Programming is Logic!

If we control the inputs, the outputs of a function should be the same, **always**. It is **completely reproducable**.

Ignore the Arithmetic

He talks a lot about basic arithmetic: feel free to ignore it, he's really just using it as a set of examples *(unless you are actually interested in the basic logical foundations of arithmetic.)*

Program Execution: Preconditions and Postconditions

We can determine a lot of the validity of a program Q by specifying some preconditions P and some postconditions R.

$$P\{Q\}R$$

If you specify enough preconditions and postconditions to be both *necessary* and *sufficient*, you have "proven" your program as correct.

Preconditions … { *Our Program* … } *Postconditions* …

Axiom of Assignment

Let's say we want to assign $x := f$. If we can assert $P(x)$ to be true after the assignment, then we must also be able to say $P(f)$ before the assignment.

$$\vdash P_0\{x := f\}P$$

We get P_0 by substituting f for x everywhere in P.

Rules of Consequence

If $\vdash P\{Q\}R$ and $\vdash R \supset S$ then $\vdash P\{Q\}S$.

If $\vdash P\{Q\}R$ and $\vdash S \supset P$ then $\vdash S\{Q\}R$.

This means we can make more general preconditions and postconditions, and they must also hold.

Example: If our output should be a positive integer, we can assert that it is just an integer.

Rule of Composition

If $\vdash P\{Q_1\}R_1$ and $\vdash R_1\{Q_2\}R$ then $\vdash P\{(Q_1; Q_2)\}R$.

This means we can chain together our statements and still make proofs!

Rule of Iteration

If $\vdash P \wedge B\{S\}P$ then $\vdash P\{\texttt{while } B \texttt{ do } S\}\neg B \wedge P$.

At the end of the while loop, the conditional B is no longer true.

Reservations and Limitations

- No side effects in the proof!
- No infinite loops!
- Only makes sense if you can rigorously assert things.

Questions?