An Axiomatic Basis for Computer Programming: Review

Gosha Krikun

Innopolis University
g.krikun@innopolis.ru

February 14, 2017

Abstract

This is review on C.A.R. Hoare's paper, An Axiomatic Basis for Computer Programming.

I. Introduction

"People are not perfect", this is what we learned so fourth. Every one could make a mistake. In some areas it could be unnoticed, but in others it could cause serious consequences.

There are a lot of people with their needs. Majority of them lie outside safety-critical sector, error and failures in which could leads to deaths. People won't spent big money for time of qualified developers to avoid minor bugs.

So nowadays main approach to find failures and errors is testing. And I can't not mentioned my favorite quote about it:

" *Program testing can be used to show the presence of bugs, but never to show their absence!* "

— Edsger Wybe Dijkstra

But more and more critical tasks entrust to computer systems. When errors and failures cost increases, we will pay for diligence and accuracy as much as needed.

To be exactly sure, that we don't make mistakes we make premises and develop techniques for reasoning about properties and correctness. And for proving it we used our logic and deduction.

II. Purpose and Applications

So Hoare suggested triple {P} Q {R} [Hoa69] in which developer need define precondition (premises) and postcondition (conclusion) which hold for program Q after execution.

This technique helps prove correctness of a program Q, according to defined logical predicates P and R.

It doesn't help derive true predicates from requirements, but it provide ability to use common proof techniques for deducting correctness.

For next disscussion suppose that we define right P and R from some requirements.

Speaking about correctness we should understand, that any answer could be in three state: answer received and it is correct, answer not received, answer is incorrect.

So when answer received and it is right, for all input values in the domain, then we call it total correctness.

If answer could be not received (e.g. program doesn't terminate) for some values in the domain, but is right for others - partial correctness.

And incorrectness in last case.

Using axioms and inference rules what was proposed by Hoare, we could deduct that if P holds before execution of Q, and if Q terminates, then R will holds after execution (correctness of Q).

In 1969 paper Hoare didn't mentioned that Q should terminates, so he defined rules just for deducting partial correctness (particular in rule D3). But later he add definition for total correctness too.

For example lets take task on finding real roots for quadratic equation.

We limits input set according to discriminant and set of real numbers in predicate P. Then we could find values by some program Q. And finally check whether equation holds or not in R.

III. Axioms and Rules

In the article Hoare states four basic rules:

- Axiom of Assignment
- Rule of Consequence
- Rule of Composition
- Rule of Iteration

Also it requires no side effects for Q (nothing changes except state which we define in pre and postconditions). Because in other case, we couldn't consider correctness of overall state.

Using this rules we can make deductive proof from our premise to conclusion, that R will holds after execution.

But only after, not during.

It could be a case when program Q during execution change state which doesn't hold conditions P and R.

This case could appear when specification describe something in general, but doesn't divide execution into small pieces of actions.

If we think about it in more details, we could decompose it to two sequence of statements (rule of composition D2), in which first part hold some postcondition R1 and second have R1 as precondition.

So nothing scary in that case if we still could prove correctness of our program.

P like a guard, filter input on what program will work correctly, and R check results to be correct.

However precondition and postconditions could be almost same. This looks ugly.

firstly we should define the truth.

So directed tests could show that implementation isn't work properly. This is helpful during development and evolving software product.

IV. Shortcomings

V. Conclusion

References

[Hoa69] C. Antony R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.