An Axiomatic Basis for Computer Programming

C.A.R. Hoare

Hoare's Axiomatic Semantics

"An Axiomatic Basis for Computer Programming", *CACM* , 1969

"Procedures and Parameters: An Axiomatic Approach," *Proceedings of the Symposium on Semantics of Algorithmic Languages*, 1971

"Proof of Correctness of Data Representations," *Acta Informatica*, 1972

An Axiomatic Basis for Computer Programming

- Provided the basis for proving programs correct with respect to their specifications
- Introduced the **P{Q}R** notation
- Based on earlier work by Floyd (1967), which was applied to flowcharts
- Presented a set of axioms for computer arithmetic

An axiomatic definition serves as a:

- Contract between a language designer and an implementer
- Reference manual for a programmer
- Axiomatic basis for formal proofs of properties of programs

Axiomatic definition comprises a deductive system

- Axioms defining the primitive constructs of the language
- Rules of inference
- Underlying logical system (e.g., first order predicate calculus with equality)

Axioms for Integer Arithmetic

- Standard arithmetic axioms
- Axioms for the finite arithmetic of computers
  - Strict interpretation
  - Firm boundary
  - Modulo arithmetic

Some Axioms for Integers

A1 x+y = y+x
A2 x*y = y*x
A3 (x+y)+z = x+(y+z)
A4 (x*y)*z = x*(y*z)
A5 x*(y+z) = x*y + x*z
A6 y ≤ x → (x-y) + y = x
A7 x+0 = x
A8 x*0 = 0
A9 x*1 = x

- Finite Arithmetic
  ∀x (x ≤ max)
- Overflow
  – strict interpretation
  ~∃x (x = max + 1)
  – firm boundary
  max +1 = max
  – modulo arithmetic
  max + 1 = 0

Strict Interpretation

+	0	1	2	3
0	0	1	2	3
1	1	2	3	
2	2	3		
3	3			

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2		
3	0	3		

Firm Interpretation

+	0	1	2	3
0	0	1	2	3
1	1	2	3	
2	2	3		
3	3			

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2		
3	0	3		

Modulo Interpretation

+	0	1	2	3
0	0	1	2	3
1	1	2	3	
2	2	3		
3	3			

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2		
3	0	3		

Partial Correctness Notation

P{Q}R

P,R are predicates

Q is a program or piece of code

If the assertion P is true before initiation of program Q, then assertion R will be true on its completion

More Notation

|- Theoremhood

R_e^x

$R_{e \rightarrow x}$

$\dfrac{H1, H2, \ldots, Hn}{Hn+1}$ whenever H1 through Hn are true Hn+1 is true

D0: Axiom of Assignment

$P_e^x\{x:=e\}P$

Example:

y>8 {x := y + 4} x>12

Hoare's Proof Technique

- Uses sentences of the form P{S}Q
- A proof of a sentence P{S}Q is a sequence of sentences the last of which is P{S}Q
- Where each sentence is:
  - an instantiation of an axiom
  - a theorem in the underlying logical system
  - follows from previous lines by applying a rule of inference

D1: Rules of Consequence

$\dfrac{P\{Q\}R,\ R \rightarrow S}{P\{Q\}S}$ $\dfrac{P\{Q\}R,\ S \rightarrow P}{S\{Q\}R}$

D2: Rule of Composition

$\dfrac{P\{Q1\}R1,\ R1\{Q2\}R}{P\{Q1;\ Q2\}R}$


```
PROCEDURE TEST (A, B: INTEGER;
                VAR X, Y, Z: INTEGER);
BEGIN
   X := A + B;
   Y := A - B;
   Z := X + Y
END;
```

ENTRY: true

EXIT: X = A + B & Y = A - B & Z = 2A

D3: Rule of Iteration

$$\frac{P \;\&\; B\{S\}P}{P\ \{\text{while } B \text{ do } S\}\ {\sim}B \;\&\; P}$$


```
PROCEDURE  FACT ( N:INTEGER; VAR Y:INTEGER);
VAR X: INTEGER;
BEGIN
   X := 0;
   Y := 1;
   ASSERT ( Y = X! & X ≤ N )
   WHILE X < N DO BEGIN
       X := X + 1;
       Y := Y * X
   END
END;
```

ENTRY: $N \geq 0$
EXIT: $Y = N!$

Procedures and Parameters: An Axiomatic Approach

- Extended the axiomatic approach to procedures
- Dealt explicitly with recursion

  "This assumption of what we want to prove before embarking on the proof explains well the aura of magic which attends a programmer's first introduction to recursive programming"

Rule of Recursive Invocation

$$\frac{p(x){:}(v)\ \text{proc}\ Q,\;\; P\{\text{call}\ p(x){:}(v)\}R\ \ |\!-\ P\{Q\}R}{P\{\text{call}\ p(x){:}(v)\}R}$$

An Axiomatic Definition of the Programming Language PASCAL

C.A.R. Hoare and N. Wirth

Acta Informatica 1973