An Axiomatic Basis for Computer Programming

Tony Hoare, 1969

Presented by Alexa VanHattum, Great Works in PL Spring 2019

Mentor Jonathan DiLorenzo

software bugs are bad

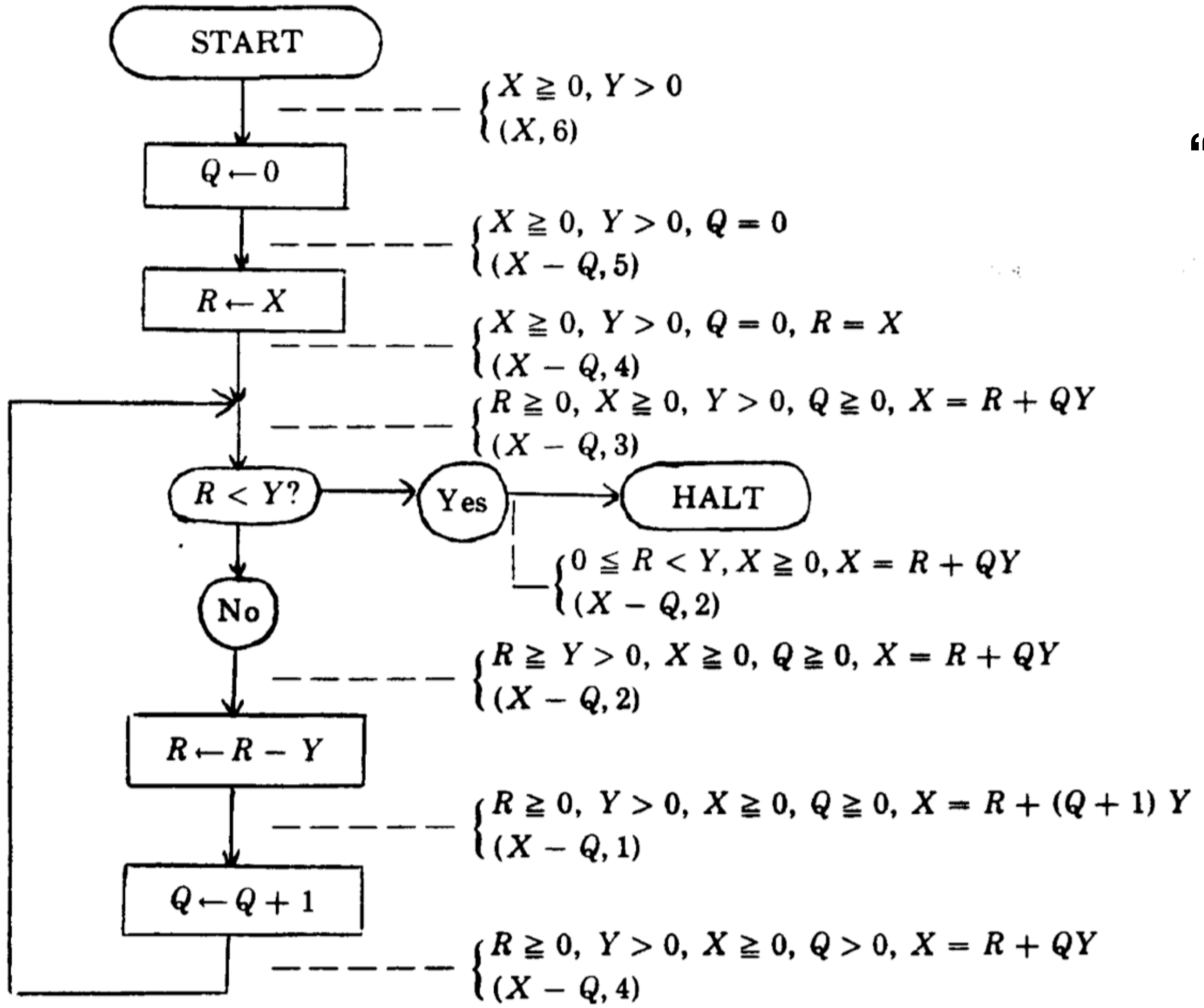
manual testing is not enough

formal reasoning is better

Motivation

"Computer programming is an exact science in that all the properties of a program and all the consequences of executing it in any given environment can, in principle, be found out from the text of the program itself by means of purely deductive reasoning."

Historical Context

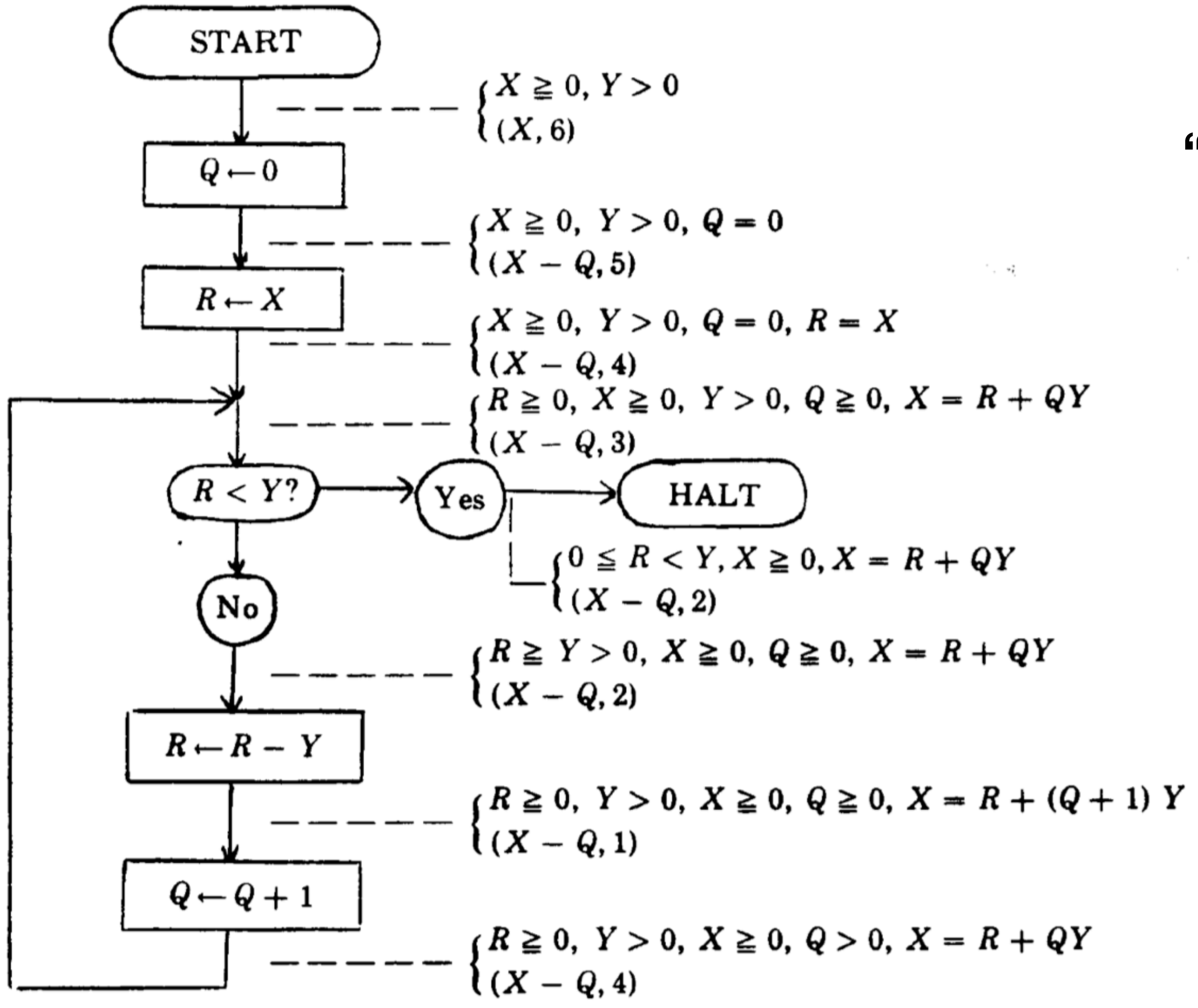


START
$X \geqq 0, Y > 0$
$(X, 6)$
$Q \leftarrow 0$
$X \geqq 0,\ Y > 0,\ Q = 0$
$(X - Q, 5)$
$R \leftarrow X$
$X \geqq 0,\ Y > 0,\ Q = 0,\ R = X$
$(X - Q, 4)$
$R \geqq 0,\ X \geqq 0,\ Y > 0,\ Q \geqq 0,\ X = R + QY$
$(X - Q, 3)$
$R < Y?$
Yes
HALT
$0 \leqq R < Y, X \geqq 0, X = R + QY$
$(X - Q, 2)$
No
$R \geqq Y > 0,\ X \geqq 0,\ Q \geqq 0,\ X = R + QY$
$(X - Q, 2)$
$R \leftarrow R - Y$
$R \geqq 0,\ Y > 0,\ X \geqq 0,\ Q \geqq 0,\ X = R + (Q + 1)\ Y$
$(X - Q, 1)$
$Q \leftarrow Q + 1$
$R \geqq 0,\ Y > 0,\ X \geqq 0,\ Q > 0,\ X = R + QY$
$(X - Q, 4)$

FIGURE 5. Algorithm to compute quotient Q and remainder R of $X \div Y$, for integers $X \geqq 0$, $Y > 0$

“Assigning Meaning to Programs”
Robert Floyd, 1967

Historical Context



START
$Q \leftarrow 0$
$R \leftarrow X$
$R < Y?$
Yes
HALT
No
$R \leftarrow R - Y$
$Q \leftarrow Q + 1$
$\{X \geqq 0, Y > 0$; $(X, 6)$
$\{X \geqq 0, Y > 0, Q = 0$; $(X - Q, 5)$
$\{X \geqq 0, Y > 0, Q = 0, R = X$; $(X - Q, 4)$
$\{R \geqq 0, X \geqq 0, Y > 0, Q \geqq 0, X = R + QY$; $(X - Q, 3)$
$\{0 \leqq R < Y, X \geqq 0, X = R + QY$; $(X - Q, 2)$
$\{R \geqq Y > 0, X \geqq 0, Q \geqq 0, X = R + QY$; $(X - Q, 2)$
$\{R \geqq 0, Y > 0, X \geqq 0, Q \geqq 0, X = R + (Q + 1) Y$; $(X - Q, 1)$
$\{R \geqq 0, Y > 0, X \geqq 0, Q > 0, X = R + QY$; $(X - Q, 4)$

FIGURE 5. Algorithm to compute quotient Q and remainder R of $X \div Y$, for integers $X \geqq 0$, $Y > 0$

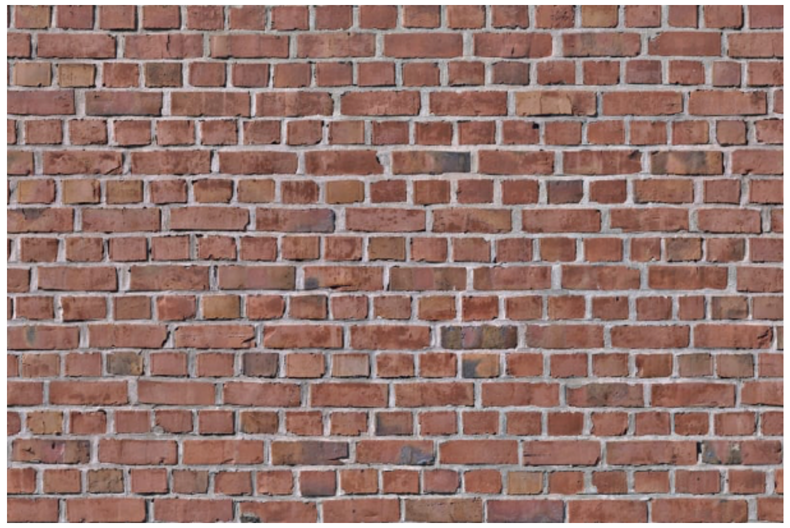
“Assigning Meaning to Programs”
Robert Floyd, 1967

“If the initial values of the program variables satisfy the relation R_1, the final values on completion will satisfy the relation R_2”

The Strategy

bra.org

economictimes.indiatimes.com


rebelwalls.com

Axioms

Deductive Rules

Theorems

Hoare’s contribution

$$P\{Q\}R$$

Precondition Program Postcondition

If P holds and Q executes and terminates, then R holds

Valid Hoare Triples?

`true {x := 1} x = 1` ✅

`x = 0 {x := x + 1} x = 1` ✅

`x = n {x := x * 2} x = 2n` ✅

`false {x := 1} x = 0` ✅

`x > 0 {while x > 1 do x := x + 1} x < 1` ❌

`x > 0 {while x > 1 do x := x + 1} x = 1` ✅

Hoare's Axioms

Integer arithmetic

A1	$x + y = y + x$
A2	$x \times y = y \times x$
A3	$(x + y) + z = x + (y + z)$
A4	$(x \times y) \times z = x \times (y \times z)$
A5	$x \times (y + z) = x \times y + x \times z$
A6	$y \leqslant x \supset (x - y) + y = x$
A7	$x + 0 = x$
A8	$x \times 0 = 0$
A9	$x \times 1 = x$

Overflow?

That depends!

1. Strict interpretation
2. Firm boundary
3. Modulo arithmetic

Assume 1 or 2 for now

how do we apply this reasoning to programs?

axiom schemas!

Assignment

$$P\{x := n\}R$$

Assignment

$$P\{x := n\}R$$

$$= P[n/x]$$ ✗

```
x = 0 {x := 1} ?
x = 0 {x := 1} (x = 0)[1/x]
x = 0 {x := 1} 1 = 0
```

✗

Assignment

$$P\{x := n\}R$$

$$= R[n/x]$$ ✅

```
          ? {x := 1} x = 1
(x = 1)[1/x] {x := 1} x = 1
      1 = 1 {x := 1} x = 1
       true {x := 1} x = 1  ✅
```

Assignment

$$\vdash R[n/x]\{x := n\}R$$

Consequence

$$\text{If } \vdash P\{Q\}R \text{ and } R \Rightarrow S,$$
$$\text{then } \vdash P\{Q\}S$$

$$\text{If } \vdash P\{Q\}R \text{ and } S \Rightarrow P,$$
$$\text{then } \vdash S\{Q\}R$$

Composition

$$\text{If } \vdash P\{Q_1\}R_1 \text{ and } \vdash R_1\{Q_2\}R,$$
$$\text{then } \vdash P\{Q_1; Q_2\}R$$

Iteration

$$\text{If } \vdash P \land B\{S\}P,$$

$$\text{then } \vdash P\{\text{while } B \text{ do } S\}\neg B \land P$$

Iteration

$$\text{If } \vdash P \wedge B\{S\}P,$$
$$\text{then } \vdash P\{\text{while } B \text{ do } S\}\neg B \wedge P$$

```
x > 0 {while x > 1 do x := x + 1} x = 1
```

Iteration

$$\text{If } \vdash P \wedge B\{S\}P,$$

$$\text{then } \vdash P\{\text{while } B \text{ do } S\}\neg B \wedge P$$

consequence rule

`¬(x > 1) ∧ x > 0 ⇒ x = 1`

$$\frac{\texttt{x > 0 \{while x > 1 do x := x + 1\} ¬(x > 1) ∧ x > 0}}{\texttt{x > 0 \{while x > 1 do x := x + 1\} x = 1}}$$

Iteration

$$\text{If } \vdash P \wedge B\{S\}P,$$
$$\text{then } \vdash P\{\text{while } B \text{ do } S\}\neg B \wedge P$$

iteration rule

$$\frac{\dfrac{\texttt{x > 0} \wedge \texttt{x > 1}\ \{\texttt{x := x + 1}\}\ \texttt{x > 0}}{\texttt{x > 0}\ \{\texttt{while x > 1 do x := x + 1}\}\ \neg(\texttt{x > 1}) \wedge \texttt{x > 0}}}{\texttt{x > 0}\ \{\texttt{while x > 1 do x := x + 1}\}\ \texttt{x = 1}}$$

Iteration

$$\text{If } \vdash P \wedge B\{S\}P,$$
$$\text{then } \vdash P\{\text{while } B \text{ do } S\}\neg B \wedge P$$

consequence rule

$$\texttt{x > 0} \wedge \texttt{x > 1} \Rightarrow \texttt{x + 1 > 0}$$

$$\dfrac{\dfrac{\dfrac{\texttt{x + 1 > 0 \{x := x + 1\} x > 0}}{\texttt{x > 0} \wedge \texttt{x > 1 \{x := x + 1\} x > 0}}}{\texttt{x > 0 \{while x > 1 do x := x + 1\}} \neg\texttt{(x > 1)} \wedge \texttt{x > 0}}}{\texttt{x > 0 \{while x > 1 do x := x + 1\} x = 1}}$$

Iteration

$$\text{If } \vdash P \wedge B\{S\}P,$$
$$\text{then } \vdash P\{\text{while } B \text{ do } S\}\neg B \wedge P$$

$$\begin{array}{c} \hspace{10em}\text{assignment rule} \\ \hline \texttt{x + 1 > 0 \{x := x + 1\} x > 0} \\ \hline \texttt{x > 0} \wedge \texttt{x > 1 \{x := x + 1\} x > 0} \\ \hline \texttt{x > 0 \{while x > 1 do x := x + 1\}} \neg\texttt{(x > 1)} \wedge \texttt{x > 0} \\ \hline \texttt{x > 0 \{while x > 1 do x := x + 1\} x = 1} \end{array}$$

Iteration

$$\text{If } \vdash P \land B\{S\}P,$$

$$\text{then } \vdash P\{\text{while } B \text{ do } S\}\neg B \land P$$

How do we find P?
Can we automate it?

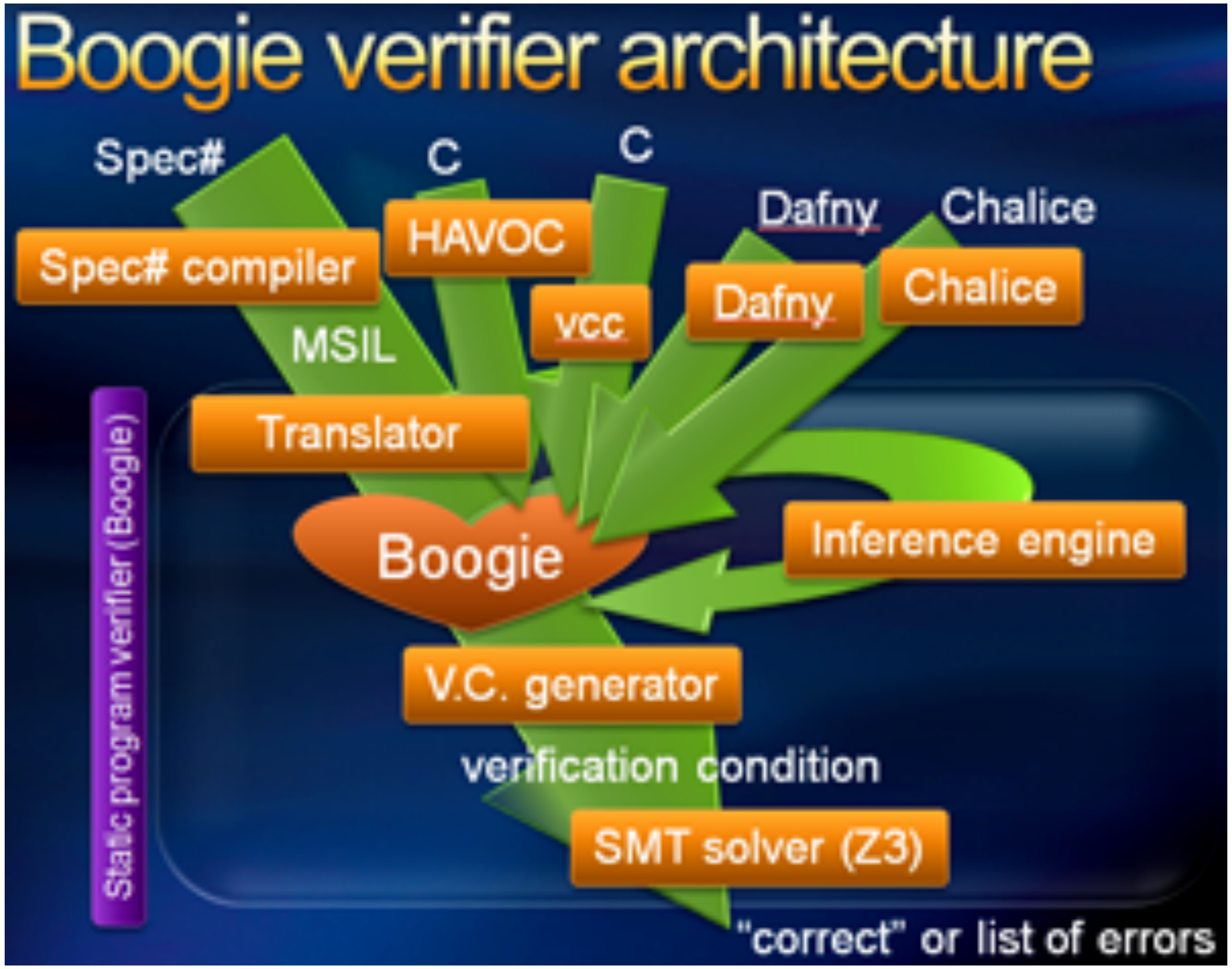
Extension to Hoare Logic: Separation Logic

- Extends Hoare logic to include reasoning over shared data
- Separation conjunction $\ast$: `P ∗ Q` asserts P and Q hold for separate regions of memory

$$\frac{\{p\}\ c\ \{q\}}{\{p \ast r\}\ c\ \{q \ast r\}}$$

The frame rule (when c does not modify the free variables of r)

Application of Hoare Logic



www.microsoft.com

Conclusion

- Relate deductive reasoning to programs via Hoare triples
- Formalize/automate axiomatic reasoning via rules
- Enable pen-and-paper proofs and automated reasoning tools
- Axioms can leave aspects of the language undefined

$$P\{Q\}R$$

Conclusion

“The practice of supplying proofs for nontrivial programs will not become widespread until considerably more powerful proof techniques become available, and even then will not be easy. But the practical advantages of program proving will eventually outweigh the difficulties, in view of the increasing costs of programming errors.”

30 years later…

“Researchers into formal methods […] predicted that the programming world would embrace with gratitude every assistance promised by formalization to solve the problems of reliability that arise when programs get large and more safety-critical […]

It has turned out that the world just does not suffer significantly from the kind of problem that our research was originally intended to solve.”

- Tony Hoare, 1996