Three Ways of Proving Termination of Loops

Krzysztof R. Apt   Frank S. de Boer   Ernst-Rüdiger Olderog

Meeting of IFIP WG 2.2 in Aachen

24–26 September 2025

Carl von Ossietzky Universität Oldenburg

Hoare's Logic [Hoa69]

Hoare's logic for **while** programs is based on correctness formulas

$$\{p\}\ S\ \{q\}$$

with the interpretation

> *"If the assertion p is true before initiation of a program S, then the assertion q will be true on its completion."*

... but program termination is originally not covered.

Approaches to proving program termination formalize Floyd's [Flo67] observation

> *"Proofs of termination are dealt with by showing that each step of a program decreases some entity which cannot decrease indefinitely."*

Termination

... continues to be a vibrant topic in program analysis, see

Annual International Termination Competition (Sept. 2025 in Leipzig)

with various competition categories, for instance proving termination of

- C programs,
- Java bytecode programs,
- logic programs,
- functional programs, and
- term rewriting systems.

Here:

- Focus on the shape of termination proofs in Hoare's logic.
- Investigate three proof rules from the proof-theoretic point of view.

Proof System I

Partial correctness of **while** programs

SKIP

$$\{p\}\ skip\ \{p\}$$

ASSIGNMENT

$$\{p[u := t]\}\ u := t\ \{p\}$$

COMPOSITION

$$\frac{\{p\}\ S_1\ \{r\},\ \{r\}\ S_2\ \{q\}}{\{p\}\ S_1\ ;\ S_2\ \{q\}}$$

CONDITIONAL

$$\frac{\{p \wedge B\}\ S_1\ \{q\},\ \{p \wedge \neg B\}\ S_2\ \{q\}}{\{p\}\ \textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}\ \{q\}}$$

LOOP I

loop invariant p

$$\frac{\{p \wedge B\}\ S\ \{p\}}{\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p \wedge \neg B\}}$$

CONS (EQUENCE)

$$\frac{p \rightarrow p_1,\ \{p_1\}\ S\ \{q_1\},\ q_1 \rightarrow q}{\{p\}\ S\ \{q\}}$$

Proof System II

Total correctness of **while** programs

To prove termination, rule LOOP I is replaced by

LOOP II bound function t

$$\frac{\begin{array}{l}\{p \wedge B\}\ S\ \{p\},\\ \{p \wedge B \wedge t = z\}\ S\ \{t < z\},\\ p \rightarrow t \geq 0\end{array}}{\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p \wedge \neg B\}}$$

where t is an integer expression such that $var(t) \subseteq var(B) \cup var(S)$ and z is an integer variable that does not appear in p, B, t or S.

LOOP II was introduced by Owicki and Gries [OG76].

B. Meyer claims: no split into invariant and bound function is needed [Mey25].

Proof System III

Total correctness of **while** programs

To prove termination, rule LOOP I is replaced by

LOOP III

$$\frac{\{p \wedge B \wedge t = z\}\ S\ \{p \wedge t < z\}, \quad p \rightarrow t \geq 0}{\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p \wedge \neg B\}}$$

where t and z are as above.

LOOP III appears in [AdBO90] and in Reynolds' book [Rey98].

Proof System IV

Total correctness of **while** programs

To prove termination, rule LOOP I is replaced by the hybrid rule

LOOP IV

$$\frac{\begin{array}{ll} \vdash_{I}\ \{p \wedge B\}\ S\ \{p\}, & \text{partial correctness} \\ \vdash_{I}\ \{p \wedge B \wedge t = z\}\ S\ \{t < z\}, & \text{partial correctness} \\ \{p \wedge B\}\ S\ \{\mathbf{true}\}, & \text{inner termination} \\ p \rightarrow t \geq 0 & \end{array}}{\{p\}\ \mathbf{while}\ B\ \mathbf{do}\ S\ \mathbf{od}\ \{p \wedge \neg B\}}$$

where t and z are as above.

LOOP IV is new.

Equivalence

We show:

The proof systems II, III, and IV are equivalent.

Two proof systems PR_1 and PR_2 are equivalent if for all formulas φ

$$\vdash_{PR_1} \varphi \quad \text{iff} \quad \vdash_{PR_2} \varphi .$$

Admissible Rules

Consider a proof system *PR*. A proof rule

$$(R) \qquad \frac{\varphi_1, \ldots, \varphi_k}{\varphi}$$

is called admissible in *PR* if

$$\vdash_{PR} \varphi_1, \ldots, \vdash_{PR} \varphi_k \quad \text{implies} \quad \vdash_{PR} \varphi.$$

So (R) does not increase the power of *PR*,
but it serves as a lemma that simplifies proofs in *PR*.

Main Theorem

Theorem 1

The loop rules are admissible in the other proof systems as follows:

1. The LOOP II rule is a admissible rule in the proof system III.
2. The LOOP III rule is a admissible rule in the proof system II.
3. The LOOP IV rule is a admissible rule in the proof system II.
4. The LOOP II rule is a admissible rule in the proof system IV.

Proof requires Lemma 1 and Theorem 2.

Corollary

The proof systems II, III, and IV are equivalent.

Auxiliary Rules

In the proof-theoretic analysis of the rules LOOP II and III, we use two auxiliary rules.

CONJ (UNCTION)

$$\frac{\{p_1\}\ S\ \{q_1\},\ \{p_2\}\ S\ \{q_2\}}{\{p_1 \wedge p_2\}\ S\ \{q_1 \wedge q_2\}}$$

∃-INTRO (DUCTION)

$$\frac{\{p\}\ S\ \{q\}}{\{\exists x : p\}\ S\ \{q\}}$$

where x does not occur in S or in $free(q)$.

Proof-theoretic Analysis

... of the rules LOOP II and III:

Lemma 1

Suppose that z is an integer variable that does not appear in p, B, t or S. Then

- $\{p \wedge B\}\ S\ \{p\},\ \{p \wedge B \wedge t = z\}\ S\ \{t < z\}$

  $\vdash_{\{\text{CONJ, CONS}\}}$

  $\{p \wedge B \wedge t = z\}\ S\ \{p \wedge t < z\}$.

- $\{p \wedge B \wedge t = z\}\ S\ \{p \wedge t < z\}$

  $\vdash_{\{\exists-\text{INTRO, CONS}\}}$

  $\{p \wedge B\}\ S\ \{p\},\ \{p \wedge B \wedge t = z\}\ S\ \{t < z\}$.

Admissibility of Auxiliary Rules

Next, we show how to dispense with the auxiliary rules.

Theorem 2

The auxiliary rules are admissible in the following proof systems:

- The CONJUNCTION rule is admissible in the proof system III.
- The $\exists$-INTRODUCTION rule is admissible in the proof system II.
- ...

Proof. Uses lemma that CONSEQUENCE rule may be applied only once, at the end. Then by induction on the structure of program S.

CONJUNCTION

$$\frac{\{p_1\}\ S\ \{q_1\},\ \{p_2\}\ S\ \{q_2\}}{\{p_1 \wedge p_2\}\ S\ \{q_1 \wedge q_2\}}$$

$\exists$-INTRODUCTION

$$\frac{\{p\}\ S\ \{q\}}{\{\exists x:\ p\}\ S\ \{q\}}$$

where x does not occur in S or in $\mathit{free}(q)$.

Proof Outlines

Proof representations in terms of annotated programs [OG76].

For LOOP III

$$\frac{\{p \wedge B \wedge t = z\}\ S\ \{p \wedge t < z\},\quad p \rightarrow t \geq 0}{\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p \wedge \neg B\}}$$

we introduce the following formation rule:

$$\frac{\{p \wedge B \wedge t = z\}\ S^*\ \{p \wedge t < z\},\quad p \rightarrow t \geq 0}{\{\textbf{inv} : p\}\{\textbf{bd} : t\}\ \textbf{while}\ B\ \textbf{do}\ \{p \wedge B \wedge t = z\}\ S^*\ \{p \wedge t < z\}\ \textbf{od}\ \{p \wedge \neg B\}}$$

where S^* is the program S annotated with some assertions, and t and z are as before.

Example Program

... involving nested loops, suggested by Tobias Nipkow:

$$
\begin{array}{l}
S_N \equiv \textbf{while } i < n \textbf{ do} \\
\qquad\qquad j := i; \\
\qquad\qquad \textbf{while } 0 < j \textbf{ do} \\
\qquad\qquad\qquad j := j - 1 \\
\qquad\qquad \textbf{od}; \\
\qquad\qquad i := i + 1 \\
\qquad \textbf{od}
\end{array}
$$

where i, j, n are integer variables.

Aim: Prove that it terminates for all initial states.

We prove $\{\textbf{true}\}\ S_N\ \{\textbf{true}\}$ in proof system III.

Proof outline for $\{\textbf{true}\}\ S_N\ \{\textbf{true}\}$ in proof system III

$$
\begin{array}{l}
\{\textbf{inv} : \textbf{true}\}\ \{\textbf{bd} : max(n-i,0)\} \\
\textbf{while } i < n \textbf{ do} \\
\quad \{\textbf{true} \wedge i < n \wedge max(n-i,0) = z_1\} \\
\quad \{n-i = z_1 \wedge z_1 > 0\} \\
\quad\ j := i; \\
\quad \{n-i = z_1 \wedge z_1 > 0\} \\
\quad \{\textbf{inv} : n-i = z_1 \wedge z_1 > 0\}\ \{\textbf{bd} : max(j,0)\} \\
\quad \textbf{while } 0 < j \textbf{ do} \\
\quad\quad \{n-i = z_1 \wedge z_1 > 0 \wedge 0 < j \wedge max(j,0) = z_2\} \\
\quad\quad \{n-i = z_1 \wedge z_1 > 0 \wedge j = z_2 \wedge z_2 > 0\} \\
\quad\quad \{n-i = z_1 \wedge z_1 > 0 \wedge j-1 < z_2 \wedge z_2 > 0\} \\
\quad\quad\ j := j-1 \\
\quad\quad \{n-i = z_1 \wedge z_1 > 0 \wedge j < z_2 \wedge z_2 > 0\} \\
\quad\quad \{n-i = z_1 \wedge z_1 > 0 \wedge max(j,0) < z_2\} \\
\quad \textbf{od}; \\
\quad \{n-i = z_1 \wedge z_1 > 0 \wedge \neg(0 < j)\} \\
\quad \{n-(i+1) < z_1 \wedge z_1 > 0\} \\
\quad\ i := i+1 \\
\quad \{n-i < z_1 \wedge z_1 > 0\} \\
\quad \{\textbf{true} \wedge max(n-i,0) < z_1\} \\
\textbf{od} \\
\{\textbf{true} \wedge \neg(i < n)\} \\
\{\textbf{true}\}
\end{array}
$$

Using Rule LOOP IV

LOOP IV

$$\vdash_{\mathrm{I}} \ \{p \wedge B\}\ S\ \{p\},$$
$$\vdash_{\mathrm{I}} \ \{p \wedge B \wedge t = z\}\ S\ \{t < z\},$$
$$\{p \wedge B\}\ S\ \{\mathbf{true}\},$$
$$p \rightarrow t \geq 0$$
$$\{p\}\ \mathbf{while}\ B\ \mathbf{do}\ S\ \mathbf{od}\ \{p \wedge \neg B\}$$

Let S be body of outer loop S_N:

$$\begin{array}{l} S \equiv j := i; \\ \quad \mathbf{while}\ 0 < j\ \mathbf{do} \\ \quad\quad j := j - 1 \\ \quad \mathbf{od}; \\ \quad i := i + 1 \end{array}$$

Proof outline in proof system I:

$$\begin{array}{l}
\{\mathbf{true} \wedge i < n \wedge max(n-i, 0) = z\} \\
\{n - i = z \wedge z > 0\} \\
\ j := i; \\
\{n - i = z \wedge z > 0\} \\
\{\mathbf{inv} : n - i = z \wedge z > 0\} \\
\quad \mathbf{while}\ 0 < j\ \mathbf{do} \\
\quad \{n - i = z \wedge z > 0\} \\
\quad\ j := j - 1 \\
\quad \{n - i = z \wedge z > 0\} \\
\mathbf{od}; \\
\{n - i = z \wedge z > 0 \wedge \neg(0 < j)\} \\
\{n - (i+1) < z \wedge z > 0\} \\
\ i := i + 1 \\
\{n - i < z \wedge z > 0\} \\
\{max(n-i, 0) < z\}
\end{array}$$

Practical Applications

Assertions annotate programs and program interfaces, e.g.,

in design by contract introduced by Bertrand Meyer for Eiffel [Mey97].

Adopted for Java with the Java Modeling Language (JML) [LCC+05] providing the designated keywords

`requires` for specifying the precondition,
`ensures` for the postcondition,
`loop_invariant`, and
`loop_decreases` for the bound function.

Annotated Java programs correspond to a proof outlines, restricted to the essential assertions for each loop (pre- and postcondition, loop invariant and bound function).

Extension to Unbounded Nondeterminism

RANDOM ASSIGNMENT (e.g. for modelling fairness [AO83])

$$\{\forall x \geq 0 : p\}\ x :=?\ \{p\}$$

LOOP II*

$$\frac{\begin{array}{l}\{p \wedge B\}\ S\ \{p\},\\ \{p \wedge B \wedge t = \alpha\}\ S\ \{t < \alpha\},\\ p \rightarrow t \geq 0\end{array}}{\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p \wedge \neg B\}}$$

where t is an expression ranging over ordinals such that $var(t) \subseteq var(B) \cup var(S)$ and α is a variable ranging over ordinals that does not appear in p, B, t or S.

Conclusion

Proof-theoretic analysis of three rule for proving the termination of loops.

Future work:

- Explore the benefits of rule LOOP IV in practice.
- Can we drop the assumption $var(t) \subseteq var(B) \cup var(S)$ in the choice of the bound function t for a loop **while** B **do** S **od** ?

References I

K. R. Apt, F. S. de Boer, and E.-R. Olderog.
Proving termination of parallel programs.
In W. H. J. Feijen, A. J. M. van Gasteren, D. Gries, and J. Misra, editors, *Beauty is Our Business, A Birthday Salute to Edsger W. Dijkstra*, pages 0–6, New York, 1990. Springer-Verlag.

K. R. Apt, F. S. de Boer, and E.-R. Olderog.
Verification of Sequential and Concurrent Programs.
Springer-Verlag, New York, third edition, 2009.

K. R. Apt, F. S. de Boer, and E.-R. Olderog.
Three ways of proving termination of loops.
In Mike Hinchey and Bernhard Steffen, editors, *The Combined Power of Research, Education, and Dissemination - Essays Dedicated to Tiziana Margaria on the Occasion of Her 60th Birthday*, volume 15240 of *LNCS*, pages 280–301. Springer, 2025.

K. R. Apt and E.-R. Olderog.
Proof rules and transformations dealing with fairness.
Sci. Comput. Program., 3(1):65–100, 1983.

R. W. Floyd.
Assigning meaning to programs.
In J. T. Schwartz, editor, *Proceedings of Symposium on Applied Mathematics 19, Mathematical Aspects of Computer Science*, pages 19–32, American Mathematical Society, New York, 1967.

C. A. R. Hoare.
An axiomatic basis for computer programming.
Commun. ACM, 12:576–580, 583, 1969.

References II

G. T. Leavens, Y. Cheon, C. Clifton, C. Ruby, and D. R. Cok.
How the design of JML accomodates both runtime assertion checking and formal verification.
Sci. of Comput. Prog., 55:185–208, 2005.

B. Meyer.
Object-Oriented Software Construction.
Prentice Hall, 2nd edition, 1997.

B. Meyer.
The nature of loops in programming.
CoRR, abs/2504.08126, 2025.

S. Owicki and D. Gries.
An axiomatic proof technique for parallel programs.
Acta Inf., 6:319–340, 1976.

J. C. Reynolds.
Theories of Programming Languages.
Cambridge University Press, Cambridge, Great Britain, 1998.