Three Ways of Proving Termination of Loops

Krzysztof R. Apt[1,2](✉), Frank S. de Boer[1](✉), and Ernst-Rüdiger Olderog[3](✉)

[1] CWI, Amsterdam, The Netherlands
{apt,F.S.de.Boer}@cwi.nl
[2] MIMUW, University of Warsaw, Warsaw, Poland
[3] Carl von Ossietzky University of Oldenburg, Oldenburg, Germany
olderog@informatik.uni-oldenburg.de

Abstract. We investigate three proof rules for proving termination of **while** programs and show their proof-theoretic equivalence. This involves a proof-theoretic analysis of various auxiliary proof rules in Hoare's logic. By discussing representations of proofs in the form of proof outlines, we reveal differences between these equivalent proof rules when used in practice. We also address applications in the context of the paradigm of design by contract.

1 Introduction

In his seminal paper [12], Hoare introduced an axiomatic method of reasoning about correctness of **while** programs, now called Hoare's logic. It is based on correctness formulas $\{p\}\ S\ \{q\}$, where S is a program and p and q are assertions (logical formulas), 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."

In this context p is referred to as a *precondition* and q as a *postcondition* of S.

However, in contrast to Floyd's earlier paper [11] that dealt with correctness of flowchart programs, program termination was not addressed. To stress this difference one distinguishes now between *partial correctness* that only focuses on the delivery of correct results, and *total correctness*, that in addition stipulates that the program terminates. So the original proposal of Hoare dealt with partial correctness.

All approaches to proving program termination within Hoare's logic formalize Floyd's [11] observation that

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

The first extension of Hoare's logic to deal with total correctness is due to [14]. Since then substantially simpler proof rules were proposed. In these proof rules variables that range over natural numbers are used. An appropriate relative completeness result, see for example [4], shows that variables ranging over more general well-founded orderings are not needed.

Termination continues to be a relevant and vibrant topic in program analysis, see for example [7] and the Annual International Termination Competition[1]. The latter comprises various competition categories, for instance proving termination of C programs, Java bytecode programs, logic programs, functional programs, and term rewriting systems. Here we focus on the shape of termination proofs in the context of Hoare's logic. To this end, we investigate three natural proof rules from the proof-theoretic point of view. More specifically, we study **while** programs with Hoare's original proof system for program correctness, in which the well-known proof rule for partial correctness of the **while** loops, which we call the LOOP I rule, is replaced by a suitable proof rule for establishing total correctness. We analyze three versions of such a proof rule:

- The LOOP II rule achieves a separation of the reasoning about the invariant and the bound function.
- The LOOP III rule allows us to document proofs as *proof outlines*, which is in particular useful when arguing about the interference freedom of component proofs in the context of parallel programs.
- The LOOP IV rule is particularly well-suited when dealing with nested loops because it modularizes the correctness proof of the outer loop from the ones of the inner loops. It is a *hybrid* proof rule with premises referring to proof systems for both partial and total correctness.

Depending on the choice of the loop rule, we obtain proof systems that we refer to by II, III, and IV, respectively. We show that these three proof systems are equivalent in the sense that every proof of a correctness formula $\{p\}\ S\ \{q\}$ carried out in one of these systems can be effectively transformed into a correctness proof in any other of these proof systems. This result is obtained by a detailed proof-theoretic analysis of the three loop rules in the context of these proof systems. To structure the proof well, we make use of auxiliary proof rules, which we show to be admissible in the proof systems.

Even though Hoare's logic has been extensively studied (see for example our survey [5]), little work has been done on the analysis of proofs in Hoare's logic. We are familiar with only three references, [2,8], and [18], in which transformations of proofs in a Hoare logic are discussed.

While these proof rules are equivalent, their use and representation in the form of proof outlines, which are programs annotated by assertions, differs. We illustrate this by analyzing a termination proof of a program with nested loops. We also address applications in the context of assertions used as annotations in the design by contract paradigm.

Dedication. We dedicate our paper to Tiziana Margaria on the occasion of her 60th birthday given that her interest in software engineering includes also meth-

[1] https://termination-portal.org/wiki/Termination_Competition.

ods for verification of software. The third author recalls various pleasant meetings at conferences and in Bremen, Passau, and Dortmund.

2 Preliminaries

Assume a given language that is determined by its set of formulas. In what follows we assume that all considered axioms, proof rules, and proof systems are concerned with the same language.

Given a proof system *PR* and two sequences of formulas $\phi_1, \ldots, \phi_m$ and $\varphi_1, \ldots, \varphi_n$ we write

$$\phi_1, \ldots, \phi_m \vdash_{PR} \varphi_1, \ldots, \varphi_n$$

to denote the fact that each formula φ_i can be proved in *PR* using as additional axioms the formulas $\phi_1, \ldots, \phi_m$. We also use this notation when the sequence $\phi_1, \ldots, \phi_m$ is empty and when *PR* is a set of proof rules.

A proof rule

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

is called ***admissible in*** *PR* if

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

Intuitively, if a rule is admissible in *PR* it does not increase the power of the proof system *PR* [18], but it serves as a lemma that simplifies proofs in *PR* by condensing a detailed proof argument into one application of (R).

We say that two proof systems PR_1 and PR_2 are ***equivalent*** if for all formulas φ

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

From now on we shall be concerned with the language, the formulas of which are either first-order formulas, called ***assertions***, or ***correctness formulas***, which are constructs of the form $\{p\}\ S\ \{q\}$, where p and q are assertions and S is a **while** program. Below we denote by $free(p)$ the set of free variables of the assertion p and by $var(t)$, $var(B)$, and $var(S)$ the set of variables that appear in the expression t, the Boolean expression B, and the program S, respectively.

We shall consider four proof systems concerned with the correctness formulas. They only differ in the used LOOP rule.

Proof system I denotes the customary proof system allowing us to prove partial correctness of **while** programs. Its axioms and proof rules, taken from our book [4], are listed in the Appendix. Its LOOP rule has the following form:

RULE LOOP I

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

In the proof system II this rule is replaced by

RULE LOOP II

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

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.

In the context of the LOOP rules discussed here, the assertion p is called the ***loop invariant*** and the expression t is called the ***bound function***. It provides an estimate how many iterations the loop will still perform before termination. The restriction $var(t) \subseteq var(B) \cup var(S)$ is added to simplify the subsequent proofs.

In the proof system III the LOOP I rule is replaced by

RULE LOOP III

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

where t and z are as above.

Finally, we shall consider the following hybrid rule that combines provability in two proof systems.

RULE LOOP IV

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

where t and z are as above.

Proof system IV is obtained from proof system I by replacing the LOOP I rule by the LOOP IV rule. The use of two forms of provability in the premises of this rule can be circumvented by the following modification of the notation. Denote the correctness formulas in the sense of partial correctness by $\{p\}\ S\ \{q\}$ and in the sense of total correctness by $[p]\ S\ [q]$. Then combine the proof system I with the proof system in which the axioms and proof rules of I except the LOOP I rule are rewritten using the $[p]\ S\ [q]$ syntax. Finally, add to this proof system the LOOP IV rewritten as follows:

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

where t and z are as above.

In what follows we use the original formulation of this rule, as it will not lead to any ambiguities.

The LOOP II rule was introduced in [16]. It corresponds to Dijkstra's modification of his weakest precondition semantics proposed in [9] and reproduced as [10]. It is difficult to determine where the LOOP III rule was introduced first. We mentioned it in [3]. It also appears in [17] on page 64 and in [1] on page 151.

The LOOP IV rule is new. It formalizes the following intuition. In order to prove the termination of a **while** loop it suffices to find a loop invariant and a bound function such that

(i) the loop invariant is maintained by each loop body execution, in the sense of partial correctness,
(ii) each loop body execution decreases the bound function, also in the sense of partial correctness,
(iii) the loop body terminates, and
(iv) the loop invariant implies that the bound function remains non-negative.

This rule supports modular reasoning about program correctness by separating the premises into partial correctness and termination properties. This is of particular relevance in the presence of nested loops, i.e., when the loop body contains inner loops. Then (i) establishes only the partial correctness of the loop body, whereas its termination is relegated to (iii). That the loop body can be iterated only finitely often is established in (ii) in combination with (iv). Note that for loop bodies without inner loops, partial and total correctness coincide. In this case the third premise, $\{p \wedge B\}\ S\ \{\mathbf{true}\}$, can then be dropped, and we arrive at the LOOP II rule. So the hybrid form of this rule arises only for nested loops.

Note that the LOOP III rule resulted from combining the first two premises of the LOOP II into one. An analogous modification can be carried out in the case of the LOOP IV rule. In what follows we disregard this possibility, given that the resulting analysis is analogous to the one concerning the LOOP II and LOOP III rules.

In the proof-theoretic analysis of the LOOP II and III rules, we make use of the following two auxiliary rules, see also [4].

RULE CONJUNCTION

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

RULE ∃-INTRODUCTION

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

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

To reason about the LOOP IV rule we shall consider the following auxiliary rule that combines provability in two proof systems.

RULE HYBRID CONJUNCTION

$$\frac{\begin{array}{l}\vdash_{\mathrm{I}} \{p_1\}\ S\ \{q_1\},\\ \{p_2\}\ S\ \{q_2\}\end{array}}{\{p_1 \wedge p_2\}\ S\ \{q_1 \wedge q_2\}}$$

A special case of this rule is the following rule from [4].

RULE DECOMPOSITION

$$\frac{\begin{array}{l}\vdash_{\mathrm{I}} \{p\}\ S\ \{q\},\\ \{p\}\ S\ \{\mathbf{true}\}\end{array}}{\{p\}\ S\ \{q\}}$$

3 Admissible Rules

The LOOP II and LOOP III rules look very much the same and it is obvious that they are in some sense equivalent. The following main theorem, that also discusses the LOOP IV rule, states this claim in the strongest way.

Theorem 1. *The loop rules are admissible in the other proof systems as follows:*

(i) The LOOP II rule is a admissible rule in the proof system III.
(ii) The LOOP III rule is a admissible rule in the proof system II.
(iii) The LOOP IV rule is a admissible rule in the proof system II.
(iv) The LOOP II rule is a admissible rule in the proof system IV.

The next lemma refers to the premises of the LOOP II and LOOP III rules. The names of the rules are abbreviated in the obvious way.

Lemma 1. *Suppose that z is an integer variable that does not appear in p, B, t or S. Then*

(i) $\{p \wedge B\}\ S\ \{p\},\ \{p \wedge B \wedge t = z\}\ S\ \{t < z\}$
$\vdash_{\{\mathrm{CONJ, CONS}\}}$
$\{p \wedge B \wedge t = z\}\ S\ \{p \wedge t < z\}.$

(ii) $\{p \wedge B \wedge t = z\}\ S\ \{p \wedge t < z\}$
$\vdash_{\{\exists-\mathrm{INTRO, CONS}\}}$
$\{p \wedge B\}\ S\ \{p\},\ \{p \wedge B \wedge t = z\}\ S\ \{t < z\}.$

Proof. (i) Immediate.

(ii) First note that by the CONSEQUENCE rule we can derive from

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

both

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

and

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

Next, by the $\exists$-INTRODUCTION rule, we derive from $\{p \wedge B \wedge t = z\}\ S\ \{p\}$

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

By the assumption about the variable z,

$$p \wedge B \rightarrow (p \wedge B \wedge \exists z : t = z) \rightarrow (\exists z : p \wedge B \wedge t = z),$$

so by the CONSEQUENCE rule, we derive $\{p \wedge B\}\ S\ \{p\}$, as desired. □

Next, we show how to dispense with the auxiliary rules. We shall need the following result, the proof of which we delay until Sect. 4.

Theorem 2. *The auxiliary rules are admissible in the following proof systems:*

(i) *The* $\exists$-INTRODUCTION *rule is admissible in the proof system* II.
(ii) *The* $\exists$-INTRODUCTION *rule is admissible in the proof system* III.
(iii) *The* CONJUNCTION *rule is admissible in the proof system* III.
(iv) *The* HYBRID CONJUNCTION *rule is admissible in the proof system* II, *that is,*
if $\vdash_{\mathrm{I}} \{p_1\}\ S\ \{q_1\}$ *and* $\vdash_{\mathrm{II}} \{p_2\}\ S\ \{q_2\}$, *then* $\vdash_{\mathrm{II}} \{p_1 \wedge p_2\}\ S\ \{q_1 \wedge q_2\}$.

We are now prepared for the proof of our main result.

Proof of Theorem 1.

(i) Suppose

$$\vdash_{\mathrm{III}} \{p \wedge B\}\ S\ \{p\},$$
$$\vdash_{\mathrm{III}} \{p \wedge B \wedge t = z\}\ S\ \{t < z\},$$

and that $p\ \rightarrow t \geq 0$ holds. By Lemma 1(i),

$$\vdash_{\mathrm{III} \cup \{\mathrm{CONJ, CONS}\}} \{p \wedge B \wedge t = z\}\ S\ \{p \wedge t < z\}.$$

By Theorem 2(iii) $\vdash_{\mathrm{III}} \{p \wedge B \wedge t = z\}\ S\ \{p \wedge t < z\}$, so by the LOOP III rule $\vdash_{\mathrm{III}} \{p\}$ **while** B **do** S **od** $\{p \wedge \neg B\}$.

(*ii*) Suppose

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

and that $p \ \rightarrow t \geq 0$ holds. By Lemma 1(*ii*),

$$\vdash_{II\,\cup\,\{\exists\text{–INTRO, CONS}\}} \{p \wedge B\}\ S\ \{p\}$$

and

$$\vdash_{II\,\cup\,\{\exists\text{–INTRO, CONS}\}} \{p \wedge B \wedge t = z\}\ S\ \{t < z\}.$$

By Theorem 2(*i*) $\vdash_{II} \{p \wedge B\}\ S\ \{p\}$ and $\vdash_{II} \{p \wedge B \wedge t = z\}\ S\ \{t < z\}$, so by the LOOP II rule $\vdash_{II} \{p\}$ **while** B **do** S **od** $\{p \wedge \neg B\}$.

(*iii*) Suppose

$$\vdash_{I} \{p \wedge B\}\ S\ \{p\},$$
$$\vdash_{I} \{p \wedge B \wedge t = z\}\ S\ \{t < z\},$$
$$\vdash_{II} \{p \wedge B\}\ S\ \{\mathbf{true}\},$$

and that $p \ \rightarrow t \geq 0$ holds. By Theorem 2(*iv*) applied twice $\vdash_{II} \{p \wedge B\}\ S\ \{p\}$ and $\vdash_{II} \{p \wedge B \wedge t = z\}\ S\ \{t < z\}$.
So by the LOOP II rule $\vdash_{II} \{p\}$ **while** B **do** S **od** $\{p \wedge \neg B\}$.

(*iv*) Suppose

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

and that $p \ \rightarrow t \geq 0$ holds. By omitting everywhere in these two proofs the second premise of the LOOP II rule whenever this rule is applied, we get $\vdash_{I} \{p \wedge B\}\ S\ \{p\}$ and $\vdash_{I} \{p \wedge B \wedge t = z\}\ S\ \{t < z\}$. Further, by the CONSEQUENCE rule $\vdash_{II} \{p \wedge B\}\ S\ \{\mathbf{true}\}$.
So by the LOOP IV rule $\vdash_{IV} \{p\}$ **while** B **do** S **od** $\{p \wedge \neg B\}$. □

Corollary 1. *The proof systems II, III, and IV are equivalent.*

Assume now that some notion of semantics of programs and assertions is given that includes the concept of a state, execution of a program, the notion of a state satisfying an assertion, and the truth of an assertion. We write then $\models \{p\}\ S\ \{q\}$ to denote the fact that every execution of S that starts in a state satisfying p terminates in a state satisfying q and say then that $\{p\}\ S\ \{q\}$ is ***true***. (Thus we are referring to ***total correctness***.) Next, we say then that a proof rule

$$\frac{\varphi_1, \ldots, \varphi_k}{\varphi}$$

is ***sound*** if $\models \varphi_1, \ldots, \models \varphi_k$ implies $\models \varphi$.

In [4] we proved that the LOOP II rule is sound, while in [17] it was proved that the LOOP III is sound. A natural question arises whether soundness of one of these rules can be directly deduced from the soundness of the other rule. This can be accomplished by modifying the claims of Lemma 1 as follows.

Lemma 2. *Suppose that z is an integer variable that does not appear in p, B, t or S. Then*

(i) If $\models \{p \wedge B\}\ S\ \{p\}$ and $\models \{p \wedge B \wedge t = z\}\ S\ \{t < z\}$,
then $\models \{p \wedge B \wedge t = z\}\ S\ \{p \wedge t < z\}$.
(ii) If $\models \{p \wedge B \wedge t = z\}\ S\ \{p \wedge t < z\}$,
then $\models \{p \wedge B\}\ S\ \{p\}$ and $\models \{p \wedge B \wedge t = z\}\ S\ \{t < z\}$.

Proof. It is a direct consequence of the fact that the proof rules used in Lemma 1 are sound. Thus each time one of these rules is applied, truth of the correctness formulas is preserved. □

Suppose now that the LOOP II rule is sound. To prove the soundness of the LOOP III rule assume that its premises are true. Then by Lemma 2(ii) the premises of the LOOP II rule are true, so by its soundness the conclusion of both rules is true. The same argument shows that soundness of the LOOP III rule implies soundness of the LOOP II rule.

We conclude this discussion by two remarks. First, notice that the last premise of each of the LOOP rules II, III and IV can also be modified, by considering in each case the implication $p \wedge B \rightarrow t \geq 0$ instead of $p \rightarrow t \geq 0$. It is easy to see that such a modification does not affect provability in the considered proof systems II, III, and IV. Indeed, an application of the original rule with a bound function t is also a valid application of the modified rule and an application of the modified rule with a bound function t can be replaced by an application of the original rule with the bound function **if** B **then** t **else** 0 **fi**.

Finally, for the rules LOOP II–IV we assumed that the bound function t satisfies the restriction $var(t) \subseteq var(B) \cup var(S)$. This allows for the proof of admissibility of the ∃-INTRODUCTION rule in Theorem 2, which in turn is used via Lemma 1 in the proof of Theorem 1 and thus in the proof of the equivalence result stated in Corollary 1. In future we will investigate how to avoid this restriction. We see that the equivalence of the different proof systems for total correctness depends very subtly on the intricate interplay of the loop rules with the admissibility of standard auxiliary rules of Hoare's logic, which requires further investigation.

4 Proof of Theorem 2

To prove this theorem we first establish two lemmas that provide additional information about the proofs in the considered proof systems.

Lemma 3. *Let PR be one of the proof systems I, II, III, or IV. If PR $\vdash \varphi$, there exists a proof of φ in PR with exactly one final application of the CONSEQUENCE rule.*

Proof. Since implication $\rightarrow$ is reflexive, we can always add to a given proof in *PR* one final application of the CONSEQUENCE rule. Since implication is transitive, successive applications of the CONSEQUENCE rule in a proof in *PR* can be condensed into one application. □

Lemma 4. *Let PR be one of the proof systems I, II, III, or IV.*

(i) Suppose that $\vdash_{PR} \{p\}\ S_1;\ S_2\ \{q\}$*. Then for some assertion* r

$$\vdash_{PR} \{p\}\ S_1\ \{r\} \text{ and } \vdash_{PR} \{r\}\ S_2\ \{q\}.$$

(ii) Suppose that $\vdash_{PR} \{p\}$ **if** B **then** S_1 **else** S_2 **fi** $\{q\}$*. Then*

$$\vdash_{PR} \{p \wedge B\}\ S_1\ \{q\} \text{ and } \vdash_{PR} \{p \wedge \neg B\}\ S_2\ \{q\}.$$

Proof. (i) By Lemma 3, the considered correctness formula was proved using the COMPOSITION rule followed by a single application of the CONSEQUENCE rule. So for some assertions p_1, r, q_1, we have

$$\vdash_{PR} \{p_1\}\ S_1\ \{r\} \text{ and } \vdash_{PR} \{r\}\ S_2\ \{q_1\}$$

and the implications $p \rightarrow p_1$ and $q_1 \rightarrow q$ hold. We now get the claim by the CONSEQUENCE rule.

(ii) The argument is analogous as in (ii). □

We now turn to the proof of Theorem 2, repeating the four statements at the beginning of each proof part.

Proof. (i) The ∃-INTRODUCTION rule is admissible in the proof system II.

We proceed by induction on the structure of S and consider a proof of $\{p\}\ S\ \{q\}$ in the proof system II, where the last two steps involve an axiom or a rule of the proof system II for the top-level operator of the program S, followed by one final application of the CONSEQUENCE rule according to Lemma 3. In all cases we assume that $x \notin var(S) \cup free(q)$.

- *Case* $S \equiv u := t$. Thus suppose $\vdash_{\mathrm{II}} \{p\}\ S\ \{q\}$. Then for some assertion q_1,

  $$\vdash_{\mathrm{II}} \{q_1[u := t]\}\ S\ \{q_1\}$$

  by the ASSIGNMENT axiom, and the implications $p \rightarrow q_1[u := t]$ and $q_1 \rightarrow q$ hold. We assumed that $x \notin var(S)$, so $x \not\equiv u$. By the assignment axiom, also

  $$\{(\exists x : q_1)[u := t]\}\ S\ \{\exists x : q_1\}.$$

  Note that the implications $\exists x : p \rightarrow \exists x : (q_1[u := t])$ and $\exists x : q_1 \rightarrow \exists x : q$ hold. Since $x \not\equiv u$ and $x \notin free(q)$, also the implications

  $$\exists x : (q_1[u := t]) \rightarrow (\exists x : q_1)[u := t] \quad\text{and}\quad (\exists x : q) \rightarrow q$$

  hold. So the CONSEQUENCE rule yields $\vdash_{\mathrm{II}} \{\exists x : p\}\ S\ \{q\}$, as desired.

- *Case* $S \equiv S_1;\ S_2$. Thus suppose $\vdash_{\rm II} \{p\}\ S\ \{q\}$. By Lemma 4($i$) for some assertion r,

$$\vdash_{\rm II} \{p\}\ S_1\ \{r\} \text{ and } \vdash_{\rm II} \{r\}\ S_2\ \{q\}.$$

Since $r \rightarrow \exists x : r$, the CONSEQUENCE rule yields $\vdash_{\rm II} \{p\}\ S_1\ \{\exists x : r\}$. Since $x \notin \mathit{free}(\exists x : r)$, the induction hypothesis yields $\vdash_{\rm II} \{\exists x : p\}\ S_1\ \{\exists x : r\}$. Since $x \notin \mathit{free}(q)$, by the induction hypothesis, also $\vdash_{\rm II} \{\exists x : r\}\ S_2\ \{q\}$. Thus by the COMPOSITION rule, $\vdash_{\rm II} \{\exists x : p\}\ S\ \{q\}$, as desired.

- *Case* $S \equiv$ **if** B **then** S_1 **else** S_2 **fi**. Thus suppose $\vdash_{\rm II} \{p\}\ S\ \{q\}$. Then by Lemma 4($ii$),

$$\vdash_{\rm II} \{p \wedge B\}\ S_1\ \{q\} \text{ and } \vdash_{\rm II} \{p \wedge \neg B\}\ S_2\ \{q\}.$$

By the induction hypothesis,

$$\vdash_{\rm II} \{\exists x : (p \wedge B)\}\ S_1\ \{q\} \text{ and } \vdash_{\rm II} \{\exists x : (p \wedge \neg B)\}\ S_2\ \{q\}.$$

Since $x \notin \mathit{var}(B)$, the CONSEQUENCE rule yields

$$\vdash_{\rm II} \{(\exists x : p) \wedge B\}\ S_1\ \{q\} \text{ and } \vdash_{\rm II} \{(\exists x : p) \wedge \neg B\}\ S_2\ \{q\}.$$

By the CONDITIONAL rule, $\vdash_{\rm II} \{\exists x : p\}\ S\ \{q\}$, as desired.

- *Case* $S \equiv$ **while** B **do** S_0 **od** . Thus suppose $\vdash_{\rm II} \{p\}\ S\ \{q\}$. By the assumption about t, we have $x \notin \mathit{var}(t)$ and without loss of generality we can assume $x \neq z$. Then for some assertion p_0 and an appropriate bound function t and variable z,

$$\vdash_{\rm II} \{p_0 \wedge B\}\ S_0\ \{p_0\},$$

$$\vdash_{\rm II} \{p_0 \wedge B \wedge t = z\}\ S_0\ \{t < z\},$$

and the implications $p \rightarrow p_0$, $p_0 \rightarrow t \geq 0$, $(p_0 \wedge \neg B) \rightarrow q$ hold.
Since $p_0 \rightarrow \exists x : p_0$, the CONSEQUENCE rule yields

$$\vdash_{\rm II} \{p_0 \wedge B\}\ S_0\ \{\exists x : p_0\}.$$

By the induction hypothesis, the assumption about x, and the CONSEQUENCE rule both

$$\vdash_{\rm II} \{(\exists x : p_0) \wedge B\}\ S_0\ \{\exists x : p_0\}$$

and

$$\vdash_{\rm II} \{(\exists x : p_0) \wedge B \wedge t = z\}\ S_0\ \{t < z\}.$$

So the LOOP II rule yields

$$\vdash_{\rm II} \{\exists x : p_0\}\ S\ \{(\exists x : p_0) \wedge \neg B\}.$$

Further, since $(p_0 \wedge \neg B) \rightarrow q$ holds and $x \notin (\mathit{free}(q) \cup \mathit{var}(B))$, also the implication $((\exists x : p_0) \wedge \neg B) \rightarrow q$ holds. So a final application of the CONSEQUENCE rule yields $\vdash_{\mathrm{II}} \{\exists x : p\}\ S\ \{q\}$, as desired.

(ii) The $\exists$-INTRODUCTION rule is admissible in the proof system III.
Again, we proceed by induction on the structure of S, but now consider a proof of $\{p\}\ S\ \{q\}$ in the proof system III, where the last two steps involve the axiom or rule of the proof system III for the top-level operator of the program S, followed by one final application of the CONSEQUENCE rule according to Lemma 3.
Except for the **while** statement, all cases are analogous, with $\vdash_{\mathrm{II}}$ replaced by $\vdash_{\mathrm{III}}$. The case of the **while** statement differs from (i) only in that one now considers just one correctness formula in the premise of the LOOP III rule instead of two. Since the details are the same, we omit them.

(iii) The CONJUNCTION rule is admissible in the proof system III.
We proceed by induction on the structure of S and for $i = 1, 2$ consider proofs of $\{p_i\}S\{q_i\}$ in the proof system III, where the last two steps involve the axiom or rule of the proof system III for the top-level operator of the program S, followed by one final application of the CONSEQUENCE rule according to Lemma 3.

- *Case* $S \equiv u := t$. Thus suppose $\vdash_{\mathrm{III}} \{p_1\}\ S\ \{q_1\}$ and $\vdash_{\mathrm{III}} \{p_2\}\ S\ \{q_2\}$. Then for some assertions q_{01} and q_{02}, by the ASSIGNMENT axiom, both

$$\vdash_{\mathrm{III}} \{q_{01}[u := t]\}\ S\ \{q_{01}\} \ \text{ and } \ \vdash_{\mathrm{III}} \{q_{02}[u := t]\}\ S\ \{q_{02}\},$$

and the implications $p_1 \rightarrow q_{01}[u := t]$, $q_{01} \rightarrow q_1$ and $p_2 \rightarrow q_{02}[u := t]$, $q_{02} \rightarrow q_2$ hold. The ASSIGNMENT axiom also yields

$$\{q_{01}[u := t] \wedge q_{02}[u := t]\}\ S\ \{q_{01} \wedge q_{02}\}.$$

By the implications $(p_1 \wedge p_2) \rightarrow (q_{01}[u := t] \wedge q_{02}[u := t])$ and $(q_{01} \wedge q_{02}) \rightarrow (q_1 \wedge q_2)$, the CONSEQUENCE rule yields

$$\vdash_{\mathrm{III}} \{p_1 \wedge p_2\}\ S\ \{q_1 \wedge q_2\},$$

as desired.

- *Case* $S \equiv S_1;\ S_2$. Thus suppose $\vdash_{\mathrm{III}} \{p_1\}\ S\ \{q_1\}$ and $\vdash_{\mathrm{III}} \{p_2\}\ S\ \{q_2\}$. Then by Lemma 4$(i)$ for some assertions r_1 and r_2

$$\vdash_{\mathrm{III}} \{p_1\}\ S_1\ \{r_1\} \ \text{ and } \ \vdash_{\mathrm{III}} \{r_1\}\ S_2\ \{q_1\},$$

$$\vdash_{\mathrm{III}} \{p_2\}\ S_1\ \{r_2\} \ \text{ and } \ \vdash_{\mathrm{III}} \{r_2\}\ S_2\ \{q_2\}.$$

By the induction hypothesis,

$$\vdash_{\mathrm{III}} \{p_1 \wedge p_2\}\ S_1\ \{r_1 \wedge r_2\} \ \text{ and } \ \vdash_{\mathrm{III}} \{r_1 \wedge r_2\}\ S_1\ \{q_1 \wedge q_2\},$$

so by the COMPOSITION rule,

$$\vdash_{\mathrm{III}} \{p_1 \wedge p_2\}\ S_1;\ S_2\ \{q_1 \wedge q_2\},$$

as desired.

- *Case* $S \equiv$ **if** B **then** S_1 **else** S_2 **fi**. Thus suppose $\vdash_{\mathrm{III}} \{p_1\}\ S\ \{q_1\}$ and $\vdash_{\mathrm{III}}$ $\{p_2\}\ S\ \{q_2\}$. Then by Lemma 4(*ii*),

$$\vdash_{\mathrm{III}} \{p_1 \wedge B\}\ S_1\ \{q_1\} \ \text{ and } \ \vdash_{\mathrm{III}} \{p_1 \wedge \neg B\}\ S_2\ \{q_1\},$$

$$\vdash_{\mathrm{III}} \{p_2 \wedge B\}\ S_1\ \{q_2\} \ \text{ and } \ \vdash_{\mathrm{III}} \{p_2 \wedge \neg B\}\ S_2\ \{q_2\}.$$

By the induction hypothesis,

$$\vdash_{\mathrm{III}} \{p_1 \wedge p_2 \wedge B\}\ S_1\ \{q_1 \wedge q_2\}$$

and

$$\vdash_{\mathrm{III}} \{p_1 \wedge p_2 \wedge \neg B\}\ S_2\ \{q_1 \wedge q_2\}.$$

So by the CONDITIONAL rule,

$$\vdash_{\mathrm{III}} \{p_1 \wedge p_2\}\ \textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}\ \{q_1 \wedge q_2\},$$

as desired.

- *Case* $S \equiv$ **while** B **do** S_0 **od** . Suppose $\vdash_{\mathrm{III}}$ $\{p_1\}\ S\ \{q_1\}$ and $\vdash_{\mathrm{III}}$ $\{p_2\}\ S\ \{q_2\}$. Then for some assertions p_{01}, p_{02}, appropriate bound functions t_1, t_2 and variables z_1, z_2,

$$\vdash_{\mathrm{III}} \{p_{01} \wedge B \wedge t_1 = z_1\}\ S_0\ \{p_{01} \wedge t_1 < z_1\},$$

$$\vdash_{\mathrm{III}} \{p_{02} \wedge B \wedge t_2 = z_2\}\ S_0\ \{p_{02} \wedge t_2 < z_2\},$$

the implications $p_{01} \rightarrow t_1 \geq 0$, $p_1 \rightarrow p_{01}$, $(p_{01} \wedge \neg B) \rightarrow q_1$ and $p_{02} \rightarrow t_2 \geq 0$, $p_2 \rightarrow p_{02}$, $(p_{02} \wedge \neg B) \rightarrow q_2$ hold.
Without loss of generality we can assume that $z_2 \notin \{z_1\} \cup \mathit{free}(p_{01})$. By the induction hypothesis,

$$\vdash_{\mathrm{III}} \{p_{01} \wedge p_{02} \wedge B \wedge t_1 = z_1 \wedge t_2 = z_2\}\ S_0\ \{p_{01} \wedge p_{02} \wedge t_1 < z_1 \wedge t_2 < z_2\}.$$

It suffices to consider one bound function, say t_1. Formally, we show this as follows. By the CONSEQUENCE rule,

$$\vdash_{\mathrm{III}} \{p_{01} \wedge p_{02} \wedge B \wedge t_1 = z_1 \wedge t_2 = z_2\}\ S_0\ \{p_{01} \wedge p_{02} \wedge t_1 < z_1\}.$$

Now, an application of the $\exists$-INTRODUCTION rule, which is admissible in the proof system III according to part (*ii*) of this theorem, followed by an application of the CONSEQUENCE rule yields

$$\vdash_{\mathrm{III}} \{p_{01} \wedge p_{02} \wedge B \wedge t_1 = z_1 \wedge \exists z_2 : t_2 = z_2\}\ S_0\ \{p_{01} \wedge p_{02} \wedge t_1 < z_1\}.$$

A further application of the CONSEQUENCE rule yields

$$\vdash_{\text{III}} \{p_{01} \wedge p_{02} \wedge B \wedge t_1 = z_1\}\ S_0\ \{p_{01} \wedge p_{02} \wedge t_1 < z_1\}.$$

Then by the LOOP III rule,

$$\vdash_{\text{III}} \{p_{01} \wedge p_{02}\}\ \textbf{while}\ B\ \textbf{do}\ S_0\ \textbf{od}\ \ \{p_{01} \wedge p_{02} \wedge \neg B\}.$$

The implications above yield $(p_1 \wedge p_2) \rightarrow (p_{01} \wedge p_{02})$ and $(p_{01} \wedge p_{02} \wedge \neg B) \rightarrow\ (q_1 \wedge q_2)$. Thus by the CONSEQUENCE rule,

$$\vdash_{\text{III}} \{p_1 \wedge p_2\}\ \textbf{while}\ B\ \textbf{do}\ S_0\ \textbf{od}\ \ \{q_1 \wedge q_2\},$$

as desired.

(iv) The HYBRID CONJUNCTION rule is admissible in the proof system II, that is, if $\vdash_{\text{I}} \{p_1\}\ S\ \{q_1\}$ and $\vdash_{\text{II}} \{p_2\}\ S\ \{q_2\}$, then $\vdash_{\text{II}} \{p_1 \wedge p_2\}\ S\ \{q_1 \wedge q_2\}$. The proof is analogous to the proof of *(iii)*. The only case that is somewhat different is the one concerned with the **while** statement. So we only deal with

- *Case* $S \equiv$ **while** B **do** S_0 **od** . Suppose $\vdash_{\text{I}} \{p_1\}\ S\ \{q_1\}$ and $\vdash_{\text{II}} \{p_2\}\ S\ \{q_2\}$. Then for some assertions p_{01}, p_{02} and an appropriate bound function t and variable z,

$$\vdash_{\text{I}} \{p_{01} \wedge B\}\ S_0\ \{p_{01}\},$$
$$\vdash_{\text{II}} \{p_{02} \wedge B\}\ S_0\ \{p_{02}\},$$
$$\vdash_{\text{II}} \{p_{02} \wedge B \wedge t = z\}\ S_0\ \{t < z\},$$

and the implications

$$p_1 \rightarrow p_{01},\ (p_{01} \wedge \neg B) \rightarrow q_1,\ p_{02} \rightarrow t \geq 0,\ p_2 \rightarrow p_{02},\ (p_{02} \wedge \neg B) \rightarrow q_2$$

hold. By the induction hypothesis,

$$\vdash_{\text{II}} \{p_{01} \wedge p_{02} \wedge B\}\ S_0\ \{p_{01} \wedge p_{02}\},$$

and by the induction hypothesis combined with the CONSEQUENCE rule,

$$\vdash_{\text{II}} \{p_{01} \wedge p_{02} \wedge B \wedge t = z\}\ S_0\ \{t < z\}.$$

So by the LOOP II rule,

$$\vdash_{\text{II}} \{p_{01} \wedge p_{02}\}\ \textbf{while}\ B\ \textbf{do}\ S_0\ \textbf{od}\ \ \{p_{01} \wedge p_{02} \wedge \neg B\}.$$

The implications above yield $(p_1 \wedge p_2) \rightarrow (p_{01} \wedge p_{02})$ and $(p_{01} \wedge p_{02} \wedge \neg B) \rightarrow\ (q_1 \wedge q_2)$. Thus by the CONSEQUENCE rule,

$$\vdash_{\text{II}} \{p_1 \wedge p_2\}\ \textbf{while}\ B\ \textbf{do}\ S_0\ \textbf{od}\ \ \{q_1 \wedge q_2\},$$

as desired.

□

5 Representing Proofs

5.1 Proof Outlines

Even though the LOOP rules II, III, and IV are equivalent in the sense of Theorem 1 or Corollary 1, they lead to different proofs of total correctness of **while** programs. The idea behind the LOOP II rule is to establish that p is a loop invariant and t is a bound function separately, while in the LOOP III rule both facts are established simultaneously. So the LOOP II rule looks more convenient when we want to strengthen a proof of partial correctness to a proof of total correctness: it suffices to establish two new premises concerned with the bound function t. In turn, the LOOP IV rule allows us to split the proof obligations even further, by identifying the property actually needed to be proved in terms of total correctness. This, as already mentioned, allows one to support modular reasoning.

However, matters change when we want to represent the proofs in the resulting proof systems II, III, and IV in a convenient form. Given that these proofs deal with structured programs, their most natural representation consists of so-called ***proof outlines***, a notion introduced in [16]. Informally, it is a proof representation in the form of a program annotated by the assertions arising from the appropriate rule applications. Such a representation is possible thanks to the fact that the proof rules are syntax directed. Proof outlines were introduced in [16] in order to reason about correctness of parallel programs, where they served to establish so-called interference freedom among the proofs of the component programs. However, they are also very useful as a representation of correctness proofs of sequential programs and, when some obvious assertions are deleted, as a program documentation.

Now, given that the LOOP II and IV rules have more than one premise consisting of a correctness formula, it is difficult to employ a single proof outline to represent a proof involving any of these rules. This is not the case with the LOOP III rule. To illustrate this point recall first that the proof outlines are defined by induction on the program structure. We only focus on the crucial formation rule concerned with the **while** statement. For the proof system I the following formation rule was used in [4]:

$$\frac{\{p \wedge B\}\ S^*\ \{p\}}{\{\mathbf{inv}: p\}\ \mathbf{while}\ B\ \mathbf{do}\ \{p \wedge B\}\ S^*\ \{p\}\ \mathbf{od}\ \ \{p \wedge \neg B\}}$$

where S^* is the program S annotated with some assertions.

For the proof system II we used in [4] the following formation rule:

$$\frac{\begin{array}{l}\{p \wedge B\}\ S^*\ \{p\},\\ \{p \wedge B \wedge t = z\}\ S^{**}\ \{t < z\},\\ p \rightarrow t \geq 0\end{array}}{\{\mathbf{inv} : p\}\{\mathbf{bd} : t\}\ \mathbf{while}\ B\ \mathbf{do}\ \{p \wedge B\}\ S^*\ \{p\}\ \mathbf{od}\ \ \{p \wedge \neg B\}}$$

where S^* and S^{**} are annotations of the program S with some assertions, t is an integer expression and z is an integer variable not occurring in p, t, B or S^*.[2]

Finally, for the proof system III we introduce the following formation rule:

$$\frac{\begin{array}{l}\{p \wedge B \wedge t = z\}\ S^*\ \{p \wedge t < z\},\\ p \rightarrow t \geq 0\end{array}}{\{\mathbf{inv} : p\}\{\mathbf{bd} : t\}\ \mathbf{while}\ B\ \mathbf{do}\ \{p \wedge B \wedge t = z\}\ S^*\ \{p \wedge t < z\}\ \mathbf{od}\ \ \{p \wedge \neg B\}}$$

where t and z are as above.

For a moment we defer the discussion of proof outlines for the proof system IV. One can easily prove by induction that each proof outline for the proof system I corresponds to a proof in this proof system. For example, if by the induction hypothesis the proof outline $\{p \wedge B\}\ S^*\ \{p\}$ corresponds to a proof of $\{p \wedge B\}\ S\ \{p\}$ in I, then the proof outline

$$\{\mathbf{inv} : p\}\ \mathbf{while}\ B\ \mathbf{do}\ \{p \wedge B\}\ S^*\ \{p\}\ \mathbf{od}\ \ \{p \wedge \neg B\}$$

corresponds to a proof of $\{p\}$ **while** B **do** S **od** $\{p \wedge \neg B\}$ in I, which is obtained by applying to $\{p \wedge B\}\ S\ \{p\}$ the LOOP I rule.

However, this property fails to hold for the proof outlines for the proof system II, because the second proof outline used in the premise of the formation rule is dropped. As a consequence, the proof outline for the **while** statement does not allow one to reconstruct the proof of $\{p\}$ **while** B **do** S **od** $\{p \wedge \neg B\}$ in the proof system II.

5.2 Proofs Using the LOOP III Rule

By contrast, each proof outline for the proof system III involving the LOOP III rule does correspond to a proof in this proof system, because all assertions used are retained. In other words, from each proof outline for the proof system III a proof in this system can be extracted.

To illustrate this point consider the following program S_N involving nested loops, suggested to us by Tobias Nipkow (private communication):

```
$S_N \equiv$ while $i < n$ do
              $j := i$;
              while $0 < j$ do
                  $j := j - 1$
```

[2] In [4], in contrast to [3], there is a typo and S^{**} is mentioned here instead of S^*.

```
          od ;
          $i := i + 1$
     od
```

where i, j, n are integer variables.

We would like to prove that it terminates for all initial states. To this end, we prove the correctness formula $\{\mathbf{true}\}\ S_N\ \{\mathbf{true}\}$ in the proof system III. In Fig. 1, we show the proof in the form of a proof outline, instantiating the corresponding formation rule for the **while** statement in proof system III with the following loop invariants and bound functions for the outer and the inner loop, respectively:

$$p \equiv \mathbf{true} \text{ and } t \equiv max(n - i, 0)\ ,$$

$$p \equiv n - i = z_1 \wedge z_1 > 0 \text{ and } t \equiv max(j, 0)\ .$$

Note that in a proof outline adjacent assertions stand for implications according to an application of the CONSEQUENCE rule. For instance, the assertion in line 3 implies that of line 4. Assignments are treated by backward substitution according to the ASSIGNMENT axiom. For instance, the assignment $i := i + 1$ in line 18 is dealt with by substituting i by $i + 1$ in the assertion in line 19, yielding the assertion in line 17.

```
1    $\{\mathbf{inv} : \mathbf{true}\}\ \{\mathbf{bd} : max(n - i, 0)\}$
2    while $i < n$ do
3        $\{\mathbf{true} \wedge i < n \wedge max(n - i, 0) = z_1\}$
4        $\{n - i = z_1 \wedge z_1 > 0\}$
5          $j := i;$
6        $\{n - i = z_1 \wedge z_1 > 0\}$
7        $\{\mathbf{inv} : n - i = z_1 \wedge z_1 > 0\}\ \{\mathbf{bd} : max(j, 0)\}$
8        while $0 < j$ do
9            $\{n - i = z_1 \wedge z_1 > 0 \wedge 0 < j \wedge max(j, 0) = z_2\}$
10           $\{n - i = z_1 \wedge z_1 > 0 \wedge j = z_2 \wedge z_2 > 0\}$
11           $\{n - i = z_1 \wedge z_1 > 0 \wedge j - 1 < z_2 \wedge z_2 > 0\}$
12             $j := j - 1$
13           $\{n - i = z_1 \wedge z_1 > 0 \wedge j < z_2 \wedge z_2 > 0\}$
14           $\{n - i = z_1 \wedge z_1 > 0 \wedge max(j, 0) < z_2\}$
15       od;
16       $\{n - i = z_1 \wedge z_1 > 0 \wedge \neg(0 < j)\}$
17       $\{n - (i + 1) < z_1 \wedge z_1 > 0\}$
18         $i := i + 1$
19       $\{n - i < z_1 \wedge z_1 > 0\}$
20       $\{\mathbf{true} \wedge max(n - i, 0) < z_1\}$
21   od
22   $\{\mathbf{true} \wedge \neg(i < n)\}$
23   $\{\mathbf{true}\}$
```

Fig. 1. Proof outline for $\{\mathbf{true}\}\ S_N\ \{\mathbf{true}\}$ in the proof system III. The line numbers have been added for reference only.

5.3 Proofs Using the LOOP IV Rule

Let us now move on to a discussion of the proofs involving the LOOP IV rule. This rule, just like the LOOP II rule, uses more than one correctness formula as a premise. As a result, it shares with the LOOP II rule the problem that it is not clear how to faithfully represent correctness proofs using a single proof outline. Indeed, each of the first three premises calls for a separate proof outline.

But it is not difficult to see that this would give rise to largely overlapping proof outlines. So, instead, we propose an alternative approach in which we replace these overlapping proof outlines referring to the proof system IV by an interrelated *set* of proof outlines in the sense of *partial correctness*, so referring to the proof system I.

For a given correctness formula $\{p\}\ S\ \{q\}$ to be proved in the proof system IV this set is defined as follows. First, we have a proof outline $\{p\}\ S^*\ \{q\}$ that employs the first formation rule given above and thus represents a proof of partial correctness of $\{p\}\ S\ \{q\}$ in I.

Next, for each occurrence of a loop **while** B **do** S_0 **od** in S, we have a proof outline $\{p_0 \wedge B \wedge t = z\}\ S_0^*\ \{t < z\}$, which represents a proof of partial correctness of $\{p_0 \wedge B \wedge t = z\}\ S_0\ \{t < z\}$ in I. Here p_0 is the invariant associated with the occurrence of the loop **while** B **do** S_0 **od** in the above proof outline $\{p\}\ S^*\ \{q\}$ and t is some bound function t such that $p_0 \rightarrow t \geq 0$. We omit the proof that existence of such a set of proof outlines in the sense of partial correctness ensures a proof of the corresponding correctness formula in the proof system IV. Intuitively, the use of the above proof outlines for *each* loop occurrence in S ensures by structural induction the third premise of the LOOP IV rule, so $\{p \wedge B\}\ S\ \{\mathbf{true}\}$ in the sense of total correctness.

We illustrate how such a set of proof outlines can be used to establish the proof of the correctness formula $\{\mathbf{true}\}\ S_N\ \{\mathbf{true}\}$ for Nipkow's program S_N in the proof system IV. We skip the trivial proof outline $\{\mathbf{true}\}\ S_N^*\ \{\mathbf{true}\}$, which corresponds to a proof of partial correctness of $\{\mathbf{true}\}\ S_N\ \{\mathbf{true}\}$. Let S_0 denote the body of the outer loop of S_N. Given the bound function $max(n-i, 0)$, Fig. 2 shows the proof outline which corresponds to a proof of partial correctness of

$$\{\mathbf{true} \wedge i < n \wedge max(n-i, 0) = z\}\ S_0\ \{max(n-i, 0) < z\},$$

assuming that the trivial invariant **true** is associated with this loop in the proof outline $\{\mathbf{true}\}\ S_N^*\ \{\mathbf{true}\}$. Note that $\mathbf{true} \rightarrow max(n-i, 0) \geq 0$.

Finally, the LOOP IV rule requires us to prove termination of S_0. This boils down to establish the termination of the inner loop. To this end, we introduce the bound function $max(j, 0)$. Note that $\mathbf{true} \rightarrow max(j, 0) \geq 0$. Figure 3 shows the proof outline which corresponds to a proof of partial correctness of

$$\{\mathbf{true} \wedge 0 < j \wedge max(j, 0) = z\}\ j := j - 1\ \{max(j, 0) < z\}.$$

6 Practical Applications

Research on program verification has entered practice most visibly by the use of assertions as annotations of programs and program interfaces that remain to be

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

Fig. 2. Proof outline for $\{\mathbf{true} \wedge i < n \wedge max(n-i,0) = z\}\ S_0\ \{max(n-i,0) < z\}$ in the proof system I.

```
$\{\mathbf{true} \wedge 0 < j \wedge max(j,0) = z\}$
$\{j - 1 < z \wedge z > 0\}$
  $j := j - 1$
$\{j < z \wedge z > 0\}$
$\{max(j,0) < z\}$
```

Fig. 3. Proof outline for $\{\mathbf{true} \wedge 0 < j \wedge max(j,0) = z\}\ j := j - 1\ \{max(j,0) < z\}$ in the proof system I.

implemented. This can be seen in the paradigm of *design by contract* introduced by Bertrand Meyer for his object-oriented programming language Eiffel [15]: program design starts with a specification in terms of assertions, the contract, against which the program is to be checked either statically by means of a proof or dynamically at runtime.

This paradigm has been adopted and extended to other programming languages, in particular to Java. The *Java Modeling Language* (JML) enriches Java with facilities for writing assertions (pre- and postconditions as well as class invariants) but also with a concept of abstract state space (using so-called model variables) [13][3]. For assertions, JML uses Java's Boolean expressions extended by universal and existential quantifiers. They are directly written into the Java source code in the form of comments starting with the symbols `//@` (so that the annotated source code may be processed by both an ordinary Java compiler and a specialized JML tool).

JML is designed to deal with the specification of Java classes, but here we focus on loops. JML provides the designated keywords `requires` for specifying the precondition, `ensures` for the postcondition, `loop_invariant`, and

[3] See also https://www.cs.ucf.edu/$\sim$leavens/JML/index.shtml.

`loop_decreases` for the bound function. To enhance readability, the pre- and postcondition as well as the loop invariant may be split into several assertions, each one stated after a separate repeated keyword[4]. An annotated Java program corresponds to a proof outline as discussed here, but restricted to the essential assertions for each loop (pre- and postcondition, loop invariant and bound function).

Also for the programming language C, a standardized specification language for C programs, called ACSL and inspired by JML, has been designed, see for instance [6][5].

In this paper, we have shown that the rules LOOP II-IV for proving termination of loops are equivalent. With respect to the number of premises LOOP III is clearly the simplest rule for proving termination. However, a proof using this rule in general requires more complex assertions because of the accumulation of the specifications of the bound functions of inner loops. This can be seen in the proof outline given in Fig. 1, in which the assertions inside the inner loop refer to both z_1 and z_2, where z_1 is the variable freezing the value of the bound function $max(n-i,0)$ of the outer loop and z_2 is the variable freezing the value of the bound function $max(j,0)$ of the inner loop. The reason is that in this proof outline we need to establish that the value of the bound function of the outer loop is not affected by the inner loop and that the value of the other bound function decreases.

The LOOP IV rule allows for a separate proof of termination of each loop, which does not require the specification of bound functions for the inner loops. Thus we have a trade off between a single complex proof and a number of simpler proofs, where complexity is measured by the size of the assertions. What works best in practice depends on the particular program structure.

Conflict of Interest. The author(s) has no competing interests to declare that are relevant to the content of this manuscript.

Appendix

The proof system I consists of the following axioms and rules:

AXIOM SKIP

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

AXIOM ASSIGNMENT

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

RULE COMPOSITION

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

[4] For an example, see https://www.openjml.org/examples/binary-search.html.
[5] See also https://frama-c.com/download/acsl-1.20.pdf.

RULE CONDITIONAL

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

RULE LOOP I

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

RULE CONSEQUENCE

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

Additionally, given an interpretation $\mathcal{I}$ for the underlying first-order language, we use as axioms all assertions that are true in $\mathcal{I}$. These assertions are used as premises in the CONSEQUENCE rule.

References

1. Almeida, J.B., Frade, M.J., Pinto, J.S., de Sousa, S.M.: Rigorous Software Development: An Introduction to Program Verification. Springer, London (2011). https://doi.org/10.1007/978-0-85729-018-2
2. Apt, K.R.: Recursive assertions and parallel programs. Acta Informatica **13**, 219–232 (1981)
3. Apt, K.R., de Boer, F.S., Olderog, E.-R.: Proving termination of parallel programs. In: Feijen, W.H.J., van Gasteren, A.J.M., Gries, D., Misra, J. (eds.) Beauty is Our Business: A Birthday Salute to Edsger W. Dijkstra, pp. 1–6. Springer, New York (1990). https://doi.org/10.1007/978-1-4612-4476-9_1
4. Apt, K.R., de Boer, F.S., Olderog, E.-R.: Verification of Sequential and Concurrent Programs, 3rd edn. Springer, New York (2009). https://doi.org/10.1007/978-1-84882-745-5
5. Apt, K.R., Olderog, E.-R.: Fifty years of Hoare's logic. Formal Aspects Comput. **31**(6), 751–807 (2019)
6. Baudin, P., et al.: The dogged pursuit of bug-free C programs: the Frama-C software analysis platform. Commun. ACM **64**(8), 56–68 (2021)
7. Cook, B., Podelski, A., Rybalchenko, A.: Proving program termination. Commun. ACM **54**(5), 88–98 (2011)
8. de Gouw, S., Rot, J.: Effectively eliminating auxiliaries. In: Ábrahám, E., Bonsangue, M., Johnsen, E.B. (eds.) Theory and Practice of Formal Methods. LNCS, vol. 9660, pp. 226–241. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30734-3_16
9. Dijkstra, E.W.: A great improvement (1976). http://www.cs.utexas.edu/users/EWD/ewd05xx/EWD573.PDF. Published as [10]
10. Dijkstra, E.W.: A great improvement. In: Dijkstra, E.W. (ed.) Selected Writings on Computing: A Personal Perspective. MCS, pp. 217–219. Springer, New York (1982). https://doi.org/10.1007/978-1-4612-5695-3_37
11. Floyd, R.: Assigning meaning to programs. In: Schwartz, J.T. (ed.) Proceedings of Symposium on Applied Mathematics 19. Mathematical Aspects of Computer Science, pp. 19–32. American Mathematical Society, New York (1967)

12. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM **12**, 576–580, 583 (1969)
13. Leavens, G.T., Cheon, Y., Clifton, C., Ruby, C., Cok, D.R.: How the design of JML accommodates both runtime assertion checking and formal verification. Sci. Comput. Program. **55**, 185–208 (2005)
14. Manna, Z., Pnueli, A.: Axiomatic approach to total correctness of programs. Acta Informatica **3**, 253–263 (1974)
15. Meyer, B.: Object-Oriented Software Construction, 2nd edn. Prentice Hall (1997)
16. Owicki, S., Gries, D.: An axiomatic proof technique for parallel programs. Acta Informatica **6**, 319–340 (1976)
17. Reynolds, J.C.: Theories of Programming Languages. Cambridge University Press, Cambridge (1998)
18. Tiuryn, J.: Hoare logic: from first-order to propositional formalism. In: Schwichtenberg, H., Steinbrüggen, R. (eds.) Proof and System-Reliability, pp. 323–340. Kluwer Academic Publishers (2002)