0

Proving Termination of Parallel Programs

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

Abstract

In Owicki and Gries [2] a well known proof method for verifying parallel programs based on the interference freedom test was introduced. We show that their extension of the method to deal with termination is incorrect and suggest two ways of repairing it.

1 Introduction

The Owicki-Gries method [2] for verifying partial correctness of parallel programs calls for finding interference free proof outlines for partial correctness of component programs. A proof outline for a partial correctness proof of $\{p\}\ S\ \{q\}$, where p and q are assertions and S is a program, is a construct of the form $\{p\}\ S^*\ \{q\}$, where S^* is the program S annotated with the assertions used in the proof of $\{p\}\ S\ \{q\}$. For example, consider the two component programs

$S_1 \equiv$ **while** $x > 0$ **do**
$y := 0;$
if $y = 0$ **then** $x := x - 1$ **else** $y := 0$ **fi**
od

and

$S_2 \equiv$ **while** $x > 0$ **do**
$y := 1;$
if $y = 1$ **then** $x := x - 1$ **else** $y := 1$ **fi**
od.

Here is a proof outline for $\{\mathbf{true}\}S_1\{\mathbf{true}\}$ (the one for $\{\mathbf{true}\}S_2\{\mathbf{true}\}$ is similar):

$\{\mathbf{true}\}$
while $x > 0$ **do**
  $\{\mathbf{true}\}$
  $y := 0;$
  $\{\mathbf{true}\}$
  if $y = 0$ **then** $\{\mathbf{true}\}$
    $x := x - 1$
    $\{\mathbf{true}\}$
  else $\{\mathbf{true}\}$
    $y := 0$
    $\{\mathbf{true}\}$
  fi
od
$\{\mathbf{true}\}$.

Proof outlines for component programs are *interference free* if the component programs do not invalidate the assertions in each others' proof outlines. In this case, the proof outlines remain valid annotations when the component programs are executed in parallel. In the above example, showing that S_2 does not invalidate the assertions of the above proof outline for S_1 requires proving the following. Let r be any assertion in the proof outline for S_1, let R be any assignment in S_2, and let $pre(R)$ be the precondition for R in the proof outline for S_2. Then the following must be proved:

$$\{r \wedge pre(R)\}\ R\ \{r\}.$$

Showing interference freedom for the proof outlines for S_1 and S_2 above is trivial, since all assertions in the proof outlines equal **true**. Hence, the proof outlines for S_1 and S_2 are interference free.

To extend the method to total correctness, Owicki and Gries proposed two steps. First, in the usual fashion, associate a bound function with each loop of each component program. A bound function is an integer expression that decreases with each loop iteration and remains non-negative. Clearly, the existence of a bound function ensures that the loop terminates when considered in isolation.

Second, to ensure termination of the parallel execution of the component programs, add the following interference freedom requirement: no component program increases a bound function of a loop of another component program.

Now consider the component programs S_1 and S_2 above. Using x as the bound function for both loops, it is clear that the additional interference freedom requirement is satisfied. And yet, it is also clear that S_1 and S_2 when executed in parallel need not terminate, for they may synchronize in such a fashion that x is never decreased. Hence, the additional interference freedom requirement proposed by Owicki and Gries is not correct.

2 A solution

The proof of total correctness of a loop requires showing that the bound function is decreased with each iteration. Formally, we can use the following proof rule motivated by Dijkstra [1] (EWD 573):

WHILE-RULE

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

where t is an integer expression and z is an integer variable that does not appear in p, t, B or S.

The first premise states that p is a loop invariant, the second that the bound function t is decreased with each iteration, and the third that if another iteration can be performed then t is positive.

If such a loop apppears in a component process, then interference freedom should require that the proof of the loop's correctness, using the above rule is not invalidated. The partial correctness proof outline already includes the necessary assertions concerning the first premise $\{p \wedge B\}\ S\ \{p\}$. However, it does not include the assertions concerning the second premise $\{p \wedge B \wedge t = z\}\ S\ \{t < z\}$. Returning to our example, it is readily seen that it is this part of the proof of the loop of component S_1 that is falsified by execution of component S_2. If the assertions from this second assumption are included in the proof outline, then the original interference freedom requirement of Owicki and Gries will suffice.

One way to achieve this is by starting from a modification of this proof rule where the first two premises are replaced with

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

and by introducing the following formation rule for a proof outline for total correctness of **while**-loops.

Definition (Proof Outline I: while-loops)

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

where t is an integer expression, z is an integer variable not occurring in p, t, B or S and $\{p \wedge B \wedge t = z\}\ S^*\ \{p \wedge t < z\}$ is a proof outline for total correctness.
□

The annotation $\{\mathbf{inv} : p\}$ represents the invariant of the loop **while** B **do** S **od**. Since the bound functions are now absorbed into the assertions, we can drop the condition for interference freedom of the bound functions and simply use the original definition of interference freedom for partial correctness.

With these changes the Owicki-Gries method for verifying total correctness of parallel programs is correct.

A drawback of the above method is that it forces us to mix the proofs of the invariance of p and of the decrease of t. The resulting proof outlines therefore become quite heavy. On the other hand this method provides a close relationship between program annotation and program execution. Since

$$\{p \wedge B\}\ z := t\ \{p \wedge B \wedge t = z\},$$

we can expand the conclusion of the above formation rule so that every **while**-loop starts with an assignment $z := t$:

$\{\mathbf{inv} : p\}$
while B **do**
$\{p \wedge B\}\ z := t\ \{p \wedge B \wedge t = z\}\ S^*\ \{p \wedge t < z\}$
od
$\{p \wedge \neg B\}$.

With this expansion, the following pleasing property of proof outlines for partial correctness holds again:

Claim Let $\{p_i\}\ S_i^*\ \{q_i\}$, $i \in \{1, \ldots, n\}$, be interference-free expanded proof outlines for total correctness and let S_i' for $i \in \{1, \ldots, n\}$ be the program resulting from S_i^* by deleting all assertions but keeping the new assignments of the form $z := t$. Consider an execution of the parallel program $[S_1' \| \ldots \| S_n']$ starting in a state satisfying $p_1 \wedge \ldots \wedge p_n$. Whenever the control in one of the component programs reaches a point annotated by an assertion, this assertion is true.

3 Another solution

Another possibility is to assume that the proof of decrease of t is of a particularly simple form, namely that for a loop body S

(i) all assignments inside S decrease t or leave it unchanged,

(ii) on each syntactically possible path through S at least one assignment decreases t.

By a *path* we mean here a possibly empty finite sequence of assignments. Sequential composition $\pi_1;\ \pi_2$ of paths π_1 and π_2 is lifted to sets Π_1 and Π_2 of paths by putting

$$\Pi_1;\ \Pi_2 = \{\pi_1;\ \pi_2 \mid \pi_1 \in \Pi_1 \text{ and } \pi_2 \in \Pi_2\}.$$

By ε we denote the empty sequence. For any path π we have $\pi;\ \varepsilon = \varepsilon;\ \pi = \pi$.

Definition Let S be a **while**-program. We define the path set of S, denoted by $path(S)$, by induction on the structure of S:

- $path(skip) = \{\varepsilon\}$,
- $path(u := t) = \{u := t\}$,
- $path(S_1;\ S_2) = path(S_1);\ path(S_2)$,
- $path(\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}) = path(S_1) \cup path(S_2)$,
- $path(\textbf{while } B \textbf{ do } S_1 \textbf{ od}) = \{\varepsilon\}$.

□

Thus, each path through S is identified with the sequence of assignments lying on it. Note that in the last clause we take into account only the case when the loop is terminated immediately. This is sufficient for establishing condition (ii) above.

We define the notion of a *proof outline for total correctness* as for partial correctness, except for the case of **while**-loops for which we use the following formation rule.

Definition (Proof Outline II: while-loops)

(1) $\{p \wedge B\}\ S^*\ \{p\}$,

(2) $\{pre(R) \wedge t = z\}\ R\ \{t \leq z\}$ for every assignment R within S,

(3) for each path $\pi \in path(S)$ there exists an assignment R in π such that $\{pre(R) \wedge t = z\}\ R\ \{t < z\}$,

(4) $p \wedge B \rightarrow t > 0$

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

where t is an integer expression and z is an integer variable not occurring in p, t, B or S^*. Here $\{p \wedge B\}\ S^*\ \{p\}$ is a proof outline for total correctness and $pre(R)$ stands for the assertion preceding R in this proof outline.
□

The annotation $\{\mathbf{bd} : t\}$ represents the bound function of the loop **while** B **do** S **od**. With this new definition of a proof outline for total correctness, the Owicki-Gries method for verifying total correctness of parallel programs is correct.

With this definition we can no longer justify the proof outlines for the component programs used in Section 1. Indeed, along the path $y := 0;\ y := 0$ of the first loop body the proposed bound function x does not decrease.

Observe that when the empty path ε is an element of $path(S)$, we cannot verify premise (3) of the above rule. Thus it may happen that we can prove total correctness of a **while**-program using the **while**-rule but are unable to record this proof as a proof outline for total correctness. An example is the program

$b :=$ **true**;
while b **do**
    if b **then** $b :=$ **false else** *skip* **fi**
od

whose termination can be easily established. This shows some limitations of the above approach to recording proofs of total correctness.

However, various parallel programs can be successfully handled in this way. For example, the bound function given in the proof of termination of program *Findpos* in Owicki and Gries [2] satisfies the more stringent conditions (2)–(4) given above. This provides a justification for their proof.

Note We discovered, when attempting to prove its soundness, that the original version of the Owicki-Gries method for proving total correctness is incorrect.
□

Acknowledgements: Detailed comments from David Gries enabled us to improve the presentation.

REFERENCES

[1] E. W. Dijkstra. *Selected Writings on Computing.* Springer-Verlag, New York, 1982.

[2] S. Owicki and D. Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, 6:319–340, 1976.

Krzysztof R. Apt,
Centre for Mathematics and Computer Science,
Kruislaan 413,
1098 SJ Amsterdam,
The Netherlands,
and
Department of Computer Sciences,
The University of Texas at Austin,
Taylor Hall 2.124,
Austin, Texas 78712–1188,
U.S.A.

Frank S. de Boer,
Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513,
5600 MB Eindhoven,
The Netherlands.

Ernst-Rüdiger Olderog,
Department of Computer Science,
University of Oldenburg,
2900 Oldenburg,
Federal Republic of Germany.