Verifying Properties of Parallel Programs: An Axiomatic Approach

Susan Owicki and David Gries
Communications of the ACM, 1976

Presented by Almog Benin
1/6/2014

Outline of talk

- Introduction
- The language
- The axioms
- Theorems
  - mutual exclusion, termination & deadlock
- Related work
- Conclusions

Introduction

- Concurrent programs becomes more popular
- It’s harder to prove their correction
  - Scheduling can be changed
  - Regular testing is not enough
- The suggested solution:
  - Axiomatic proof system

The language

- Derived from Algol 60
- Contains the usual statements:
  - Assignment
  - Conditional
  - Loops: while \ for
  - Compound \ null

The language – cont'

- Denote by:
  - r – a set of variables
  - S – a statement
  - B – a boolean condition
- Parallel execution statement:

  resource $r_1,\ldots,r_m$:

  cobegin $S_1 // \ldots // S_n$ **coend**
- Critical section statement:

  with r **when** B **do** S

Example

$\{x = 0\}$

begin $y \coloneqq 0, z \coloneqq 0;$

$\{y = 0 \wedge z = 0 \wedge I(r)\}$

resource $r(x, y, z)$**: cobegin**

$\{y = 0\}$

with r **when** $true$ **do**

$\{y = 0 \wedge I(r)\}$

begin $x \coloneqq x + 1; y \coloneqq 1$ **end**

$\{y = 1 \wedge I(r)\}$

$\{y = 1\}$

//

$\{z = 0\}$

with r **when** $true$ **do**

$\{z = 0 \wedge I(r)\}$

begin $x \coloneqq x + 1;$ z $\coloneqq 1$ **end**

$\{z = 1 \wedge I(r)\}$

$\{z = 1\}$

coend

$\{y = 1 \wedge z = 1 \wedge I(r)\}$

end

$\{x = 2\}$

$I(r) = \{x = y + z\}$

- Each statement has:
  - *Pre-condition P*
  - *Post-condition Q*
- *Wrote as* $\{P\}\ S\ \{Q\}$
- We assume that sequential execution is simple to be proved.
- $I(r)$ – the invariant for the resource r
  - Remains true at all times outside critical sections for r

Example – cont'

$\{x = 0\}$

begin $y := 0, z := 0;$

$\{y = 0 \land z = 0 \land I(r)\}$

resource $r(x, y, z)$: **cobegin**

$\{y = 0\}$

with r **when** $true$ **do**

$\{y = 0 \land I(r)\}$

begin $x := x + 1; y := 1$ **end**

$\{y = 1 \land I(r)\}$

$\{y = 1\}$

//

$\{z = 0\}$

with r **when** $true$ **do**

$\{z = 0 \land I(r)\}$

begin $x := x + 1$; z $:= 1$ **end**

$\{z = 1 \land I(r)\}$

$\{z = 1\}$

coend

$\{y = 1 \land z = 1 \land I(r)\}$

end

$\{x = 2\}$

$I(r) = \{x = y + z\}$

- Each statement has:
  - *Pre-condition P*
  - *Post-condition Q*
- *Wrote as* $\{P\}\ S\ \{Q\}$
- We assume that sequential execution is simple to be proved.
- $I(r)$ – the invariant for the resource r
  - Remains true at all times outside critical sections for r

Axiom #1

$\{x = 0\}$

begin $y := 0, z := 0;$

$\{y = 0 \wedge z = 0 \wedge I(r)\}$

resource $r(x, y, z)$**: cobegin**

$\{\boldsymbol{y = 0}\}$

with r **when** $true$ **do**

$\{y = 0 \wedge I(r)\}$

begin $x := x + 1; y := 1$ **end**

$\{y = 1 \wedge I(r)\}$

$\{\boldsymbol{y = 1}\}$

//

$\{\boldsymbol{z = 0}\}$

with r **when** $true$ **do**

$\{z = 0 \wedge I(r)\}$

begin $x := x + 1;$ z $:= 1$ **end**

$\{z = 1 \wedge I(r)\}$

$\{\boldsymbol{z = 1}\}$

coend

$\{y = 1 \wedge z = 1 \wedge I(r)\}$

end

$\{x = 2\}$

$I(r) = \{x = y + z\}$

- The critical section axiom:
  - If:
    - $\{I(r) \wedge P \wedge B\}\, S\, \{I(r) \wedge Q\}$
    - $I(r)$ is the invariant from the cobegin statement
    - No variable free in P or Q is changed in another thread
  - Then:
    - $\{P\}$ **with** r **when** B **do** S $\{Q\}$
- For example, set:
  - $P =$ "y $= 0$"
  - $Q = "y = 1"$
  - $B = true$

Axiom #2

$\{x = 0\}$

begin $y := 0, z := 0;$

$\{\boldsymbol{y = 0 \land z = 0 \land I(r)}\}$

resource $r(x, y, z)$: **cobegin**

$\{y = 0\}$

with r **when** $true$ **do**

$\{y = 0 \land I(r)\}$

begin $x := x + 1; y := 1$ **end**

$\{y = 1 \land I(r)\}$

$\{y = 1\}$

//

$\{z = 0\}$

with r **when** $true$ **do**

$\{z = 0 \land I(r)\}$

begin $x := x + 1$; z $:= 1$ **end**

$\{z = 1 \land I(r)\}$

$\{z = 1\}$

coend

$\{\boldsymbol{y = 1 \land z = 1 \land I(r)}\}$

end

$\{x = 2\}$

$I(r) = \{x = y + z\}$

- The parallel execution axiom:
  - If:
    - $\{P_1\}\ S_1\ \{Q_1\}$ … $\{P_n\}\ S_n\ \{Q_n\}$
    - No variable free in P_i or Q_i is changed in S_j $(i \neq j)$
    - All variables in $I(r)$ belong to resource r
  - Then:
    - $\{P_1 \land \cdots \land P_n \land I(r)\}$ **resource** r: **cobegin** $S_1 // \cdots // S_n$ **coend** $\{Q_1 \land \cdots \land Q_n \land I(r)\}$
- For example, set:
  - $P_1 =$ "$y = 0$"
  - $P_2 =$ "$z = 0$“
  - $Q_1 =$ "$y = 1$"
  - $Q_2 =$ "$z = 1$"

The consequence

$\{x = 0\}$

begin $y := 0, z := 0;$

$\{y = 0 \wedge z = 0 \wedge I(r)\}$

resource $r(x, y, z)$: **cobegin**

$\{y = 0\}$

with r **when** $true$ **do**

$\{y = 0 \wedge I(r)\}$

begin $x := x + 1; y := 1$ **end**

$\{y = 1 \wedge I(r)\}$

$\{y = 1\}$

//

$\{z = 0\}$

with r **when** $true$ **do**

$\{z = 0 \wedge I(r)\}$

begin $x := x + 1$; z $:= 1$ **end**

$\{z = 1 \wedge I(r)\}$

$\{z = 1\}$

coend

$\{y = 1 \wedge z = 1 \wedge I(r)\}$

end

$\{\boldsymbol{x = 2}\}$

$I(r) = \{x = y + z\}$

- Using the invariant, we have the result:
  - $\boldsymbol{x = 2}$

Axiom #3 - Auxiliary Variable Axiom:

resource r(x): **cobegin**

with r **when** true **do**

$x := x + 1$

//

with r **when** true **do**

$x := x + 1$

coend

- Unable to proof using the existing axioms.
- This program does the same as the former.

Axiom #3 - Auxiliary Variable Axiom:

resource r(x): **cobegin**

with r **when** true **do**

$x := x + 1$

//

with r **when** true **do**

$x := x + 1$

coend

- The solution: make use of auxiliary variables
  - Auxiliary variable is a variable which is assigned, but never used
  - Removing this variable doesn't change the program.

$y := 1$

Axiom #3 - Auxiliary Variable Axiom:

resource r(x): **cobegin**

    with r **when** true **do**

        $x \coloneqq x + 1$

//

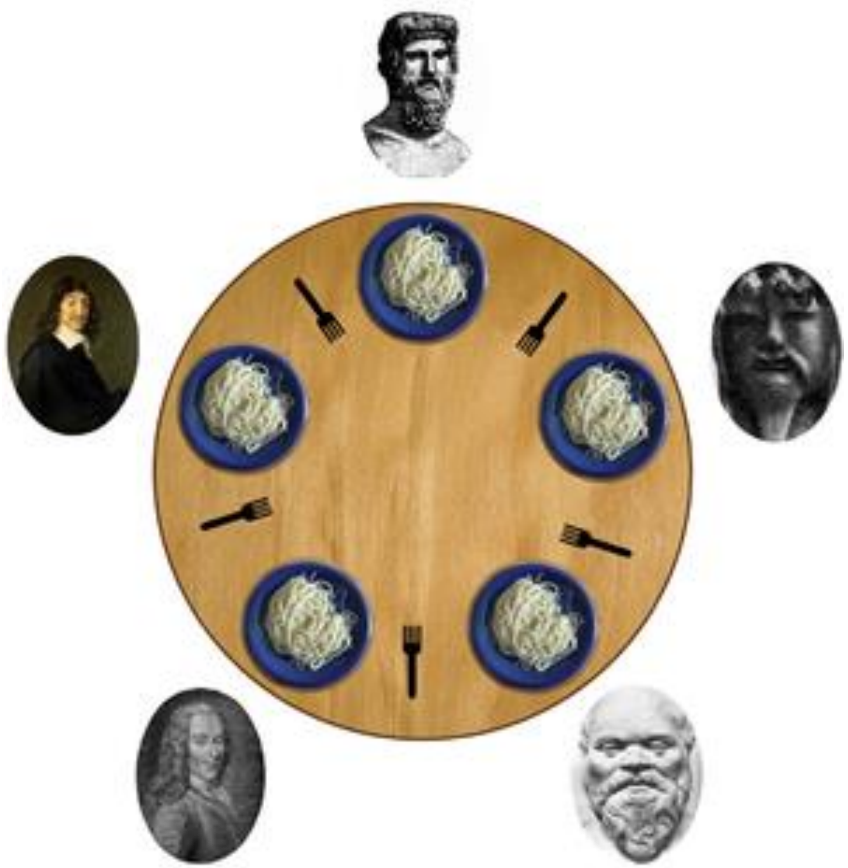
    with r **when** true **do**

        $x \coloneqq x + 1$

coend

- If:
  - AV is an auxiliary variable set for a statement S.
  - S' obtained by deleting all assignments to variables in AV.
  - $\{P\}\ S\ \{Q\}$ is true
  - P and Q don't refer to variable any variables from AV.
- Then:
  - $\{P\}\ S'\ \{Q\}$ is also true.

The Dining Philosophers Problem



- 5 bowls of spaghetti
- 5 forks
- 5 philosophers
- Each philosopher repeatedly eats and then thinks
- Needs 2 forks for eating

begin

for $j := 0$ **step** 1 **until** 4

begin af[j] := 2; $eating[j] := 0$ **end**

$\{I(forks) \wedge eating[i] = 0, 0 \le i \le 4\}$

resource $forks$: **cobegin**

$DP_0 // \cdots // DP_4$

coend

$\{I(forks) \wedge eating[i] = 0, 0 \le i \le 4\}$

end

$eating$ – an auxiliary variable

$forks := af \;\&\; eating$

DP_i:

$\{eating[i] = 0\}$

for $j := 1$ **step** 1 **until** N_i

begin

with $forks$ **when** $af[i] = 2$ **do**

$\{eating[i] = 0 \wedge af[i] = 2 \wedge I(forks)\}$

begin $af[i \ominus 1]--;\ af[i \oplus 1]--;\ eating[i] = 1$ **end**

$\{eating[i] = 1 \wedge I(forks)\}$

<eat i>

with $forks$ **when** true **do**

$\{eating[i] = 1 \wedge I(forks)\}$

begin $af[i \ominus 1]++;\ af[i \oplus 1]++; eating[i] = 0$ **end**

$\{eating[i] = 0 \wedge I(forks)\}$

<think i>

end

$\{eaing[i] = 0\}$

$$I(forks) = \left\{ \begin{bmatrix} 0 \le eating[i] \le 1 \wedge (eating[i] = 1 \Rightarrow af[i] = 2) \wedge \\ af[i] = 2 - (eating[i \ominus 1] + eating[i \oplus 1]) \end{bmatrix} 0 \le i \le 4 \right\}$$

begin

for $j := 0$ **step** 1 **until** 4

begin af$[j] := 2$; $eating[j] := 0$ **end**

$\{I(forks) \wedge eating[i] = 0, 0 \le i \le 4\}$

resource $forks$: **cobegin**

$DP_0 // \cdots // DP_4$

coend

$\{I(forks) \wedge eating[i] = 0, 0 \le i \le 4\}$

end

eating – an auxiliary variable

$forks := af \;\&\; eating$

DP_i:

$\{eating[i] = 0\}$

for $j := 1$ **step** 1 **until** N_i

begin

with $forks$ **when** $af[i] = 2$ **do**

$\{eating[i] = 0 \wedge af[i] = 2 \wedge I(forks)\}$

begin $af[i \ominus 1] --$; $af[i \oplus 1] --$; $eating[i] = 1$ **end**

$\{eating[i] = 1 \wedge I(forks)\}$

<eat *i*>

with $forks$ **when** true **do**

$\{eating[i] = 1 \wedge I(forks)\}$

begin $af[i \ominus 1] ++$; $af[i \oplus 1] ++$; $eating[i] = 0$ **end**

$\{eating[i] = 0 \wedge I(forks)\}$

<think *i*>

end

$\{eaing[i] = 0\}$

$$I(forks) = \left\{ \left[\begin{array}{c} 0 \le eating[i] \le 1 \wedge (eating[i] = 1 \Rightarrow af[i] = 2) \wedge \\ af[i] = 2 - (eating[i \ominus 1] + eating[i \oplus 1]) \end{array} \right] \middle| \; 0 \le i \le 4 \right\}$$

Mutual Exclusion

- We will prove that there are no 2 neighbors eating together.
- Assume by contradiction that there is i for which $eating[i] = eating[i \oplus 1] = 1$.
- We will derive a contradiction:
  - $\left(eating[i] = 1 \wedge eating[i \oplus 1] = 1 \wedge I(forks)\right) \Rightarrow (af[i] = 2 \wedge af[i] < 2) \Rightarrow false$
  - For that, we need to proof a new theorem.
    - Because another philosopher may be in a critical section ($I(r)$ will not hold).

Theorem #1: The Mutual Exclusion Theorem

Suppose:
- S_1 and S_2 are statements in different parallel threads of a program S
- Neither S_1 nor S_2 belongs to a critical section for resource r.
- Let P_1 and P_2 be assertions that holds during the execution of S_1 and S_2, respectively.
- $(P_1 \wedge P_2 \wedge I(r)) \Rightarrow false$

Then:

<u>S_1 and S_2 are mutually exclusive</u>

if P_1 and P_2 are true when the execution of S begins.

Example:
- $S_1 =< eat\ i >$
- $S_2 =< eat\ i \oplus 1 >$
- $P_1 = \{eating[i] = 1\}$
- $P_2 = \{eating[i \oplus 1] = 1\}$

By theorem #1:

$$(P_1 \wedge P_2 \wedge I(r)) \Rightarrow false$$

In contradiction.

Deadlock

- A **thread** is **blocked** if it is stopped at the statement **with** r **when** B **do** S because B is false or because another thread is using resource r.
- A **parallel program** is **blocked** if at least one thread is blocked, and all other threads are either finished or blocked as well.
- A parallel program is **deadlock-free** if there is no computation lead it to be blocked.

Theorem #2: The Blocking Theorem

- Suppose program S contains the statement:
  - S' = **resource** r; **cobegin** $S_1 // \cdots // S_n$ **coend**
- Let the **with-when** statements of thread S_k be
  - S_k^j = **with** r_k^j **when** B_k^j **do** T_k^j
- Let pre(S_k^j) and $I(r)$ be assertions derived from a proof of $\{P\}\ S\ \{Q\}$.

Theorem #2: The Blocking Theorem – cont’

- D_1 means: “for each thread, it is either finished or blocked at the beginning of one of its **with-when** statement”:

$$D_1 = \bigwedge_k \left\{ post(S_k) \vee \left[\bigvee_j \left(\neg B_k^j \wedge pre\left(S_k^j\right) \right) \right] \right\}$$

- D_2 means: “There is at least one thread that is blocked by one of the **with-when** statement”:

$$D_2 = \bigvee_k \bigvee_j \left(\neg B_k^j \wedge pre\left(S_k^j\right) \right)$$

<u>Then if $D_1 \wedge D_2 \wedge I(r) \Rightarrow false$, S is deadlock-free if P is true when execution begins.</u>

The Blocking Theorem in The Dining Philosophers

Let $S_i = DP_i$. Thus:

$$D_1 = \bigwedge_i eating[i] = 0 \qquad\qquad D_2 = \bigvee_i \exists i(af[i] \neq 2)$$

$$I(forks) = \left\{ \begin{bmatrix} 0 \le eating[i] \le 1 \land (eating[i] = 1 \Rightarrow af[i] = 2) \land \\ af[i] = 2 - (eating[i \ominus 1] + eating[i \oplus 1]) \end{bmatrix} 0 \le i \le 4 \right\}$$

$$D_1 \land D_2 \land I(forks) \Rightarrow false$$

And thus the dining philosophers program is deadlock free.

Theorem #3: Termination

- **Definition**: A statement T terminates conditionally if it can be proved to terminate under the assumption that it doesn't become blocked.
- **Theorem**: if T is a **cobegin** statement in a program S which is deadlock-free, T terminates if each of its parallel threads terminates conditionally.
- Easy to be proved for the dining philosophers

Related work

- This work uses a language presented by Hoare (1972).
  - However, Hoare’s solution provides partial correctness (a program that produces the correct result or doesn’t terminate).
  - It also fails to prove partial correctness for some simple programs.

Conclusion

- We presented an axiomatic proof system for parallel programs.
- We defined some theorems based on these axioms.
- We applied these theorems on the dinning philosophers problem.

Questions?

?

My thoughts

- Is that proof system cost-effective?
  - Better than normal testing?
- What is the best way to use this proof system?
  - Manual?
  - Automatic?
  - Interactive?