

Expressive Modular Fine-Grained Concurrency Specification

Bart Jacobs and Frank Piessens

DistriNet Research Group, Department of Computer Science,
Katholieke Universiteit Leuven, Belgium

POPL 2011 – Thursday, January 27, 2011, 11h30

The Bottom Line

We propose an approach
for
specifying and verifying fine-grained concurrent modules
that is
more powerful* and simpler*
but not necessarily easier to use
than
the best existing approaches
which are
Cave
(a combination of linearizability, rely-guarantee, and separation logic)
and
Concurrent Abstract Predicates

Contents

- 1 Refresher: Resource Invariants and Auxiliary Variables
 - Resource Invariants à la Hoare
 - Resource Invariants and Auxiliary Variables à la Owicki-Gries
- 2 The Procedure-Modularity Problem
- 3 Our Solution
- 4 Final Words
 - Dynamic Example: A Queue
 - More Goodies
 - Conclusion

Contents

- 1 Refresher: Resource Invariants and Auxiliary Variables
 - Resource Invariants à la Hoare
 - Resource Invariants and Auxiliary Variables à la Owicki-Gries
- 2 The Procedure-Modularity Problem
- 3 Our Solution
- 4 Final Words
 - Dynamic Example: A Queue
 - More Goodies
 - Conclusion

Contents

- 1 Refresher: Resource Invariants and Auxiliary Variables
 - Resource Invariants à la Hoare
 - Resource Invariants and Auxiliary Variables à la Owicki-Gries
- 2 The Procedure-Modularity Problem
- 3 Our Solution
- 4 Final Words
 - Dynamic Example: A Queue
 - More Goodies
 - Conclusion

A Simple Parallel Program

```
resource  $r(x)$  : cobegin  
    with  $r$  do  $x := x + 1$   
    //  
    with  $r$  do  $x := x + 1$   
coend
```

A Simple Parallel Program, with Incomplete Postcondition

```
{x = 0}  
resource r(x) : cobegin  
    with r do x := x + 1  
    //  
    with r do x := x + 1  
coend  
{0 ≤ x}
```

Hoare's Proof Rules

$$\frac{r(\bar{x}) : I \vdash \{P_1\} c_1 \{Q_1\} \quad r(\bar{x}) : I \vdash \{P_2\} c_2 \{Q_2\}}{\{P_1 \wedge P_2 \wedge I\} \text{ resource } r(\bar{x}) : \text{cobegin } c_1 // c_2 \text{ coend } \{Q_1 \wedge Q_2 \wedge I\}}$$

$$\frac{\{P \wedge I\} c \{Q \wedge I\}}{r(\bar{x}) : I \vdash \{P\} \text{ with } r \text{ do } c \{Q\}}$$

Hoare's Proof Rules, with Conditions on Variable Occurrences

$$\frac{\bar{y}; r(\bar{x}) : I \vdash \{P_1\} c_1 \{Q_1\} \quad \bar{z}; r(\bar{x}) : I \vdash \{P_2\} c_2 \{Q_2\} \quad \bar{y} \cap \bar{z} = \emptyset \quad \bar{y}\bar{z} \cap \bar{x} = \emptyset \quad \text{FV}(I) \subseteq \bar{x}}{\{P_1 \wedge P_2 \wedge I\} \text{ resource } r(\bar{x}) : \text{cobegin } c_1 // c_2 \text{ coend } \{Q_1 \wedge Q_2 \wedge I\}}$$

$$\frac{\bar{y}\bar{x} \vdash \{P \wedge I\} c \{Q \wedge I\}}{\bar{y}; r(\bar{x}) : I \vdash \{P\} \text{ with } r \text{ do } c \{Q\}}$$

Example Program, Rearranged

```
resource  $r(x)$  : cobegin
  with  $r$  do
begin  $x := x + 1$  end
//
begin  $x := x + 1$  end
coend
```

Proof of Incomplete Postcondition

$$\begin{array}{c}
 \{x = 0\} \\
 \{\mathbf{true} \wedge \mathbf{true} \wedge I\} \\
 \mathbf{resource} \ r(x) : \mathbf{cobegin} \\
 \begin{array}{cc}
 \{\mathbf{true}\} & \{\mathbf{true}\} \\
 \mathbf{with} \ r \ \mathbf{do} & \mathbf{with} \ r \ \mathbf{do} \\
 \{\mathbf{true} \wedge I\} & \{\mathbf{true} \wedge I\} \\
 \mathbf{begin} \ x := x + 1 \ \mathbf{end} & // \ \mathbf{begin} \ x := x + 1 \ \mathbf{end} \\
 \{\mathbf{true} \wedge I\} & \{\mathbf{true} \wedge I\} \\
 \{\mathbf{true}\} & \{\mathbf{true}\}
 \end{array} \\
 \mathbf{coend} \\
 \{\mathbf{true} \wedge \mathbf{true} \wedge I\} \\
 \{0 \leq x\} \\
 I \equiv 0 \leq x
 \end{array}$$

Contents

- 1 Refresher: Resource Invariants and Auxiliary Variables
 - Resource Invariants à la Hoare
 - Resource Invariants and Auxiliary Variables à la Owicki-Gries
- 2 The Procedure-Modularity Problem
- 3 Our Solution
- 4 Final Words
 - Dynamic Example: A Queue
 - More Goodies
 - Conclusion

The Example Program, with Complete Postcondition

```
{x = 0}  
resource r(x) : cobegin  
    with r do x := x + 1  
    //  
    with r do x := x + 1  
coend  
{x = 2}
```

Example Program, with Auxiliary Variables

```
      y := 0; z := 0;  
  resource r(x, y, z) : cobegin  
    with r do  
begin x := x + 1; y := 1 end //    with r do
```

Proof of Complete Postcondition

$$\begin{array}{c}
 \{x = 0\} \\
 y := 0; z := 0; \\
 \{y = 0 \wedge z = 0 \wedge I\} \\
 \text{resource } r(x, y, z) : \text{cobegin } I \equiv x = y + z \\
 \begin{array}{cc}
 \{y = 0\} & \{z = 0\} \\
 \text{with } r \text{ do} & \text{with } r \text{ do} \\
 \{y = 0 \wedge I\} & \{z = 0 \wedge I\} \\
 \text{begin } x := x + 1; y := 1 \text{ end} & // \text{begin } x := x + 1; z := 1 \text{ end} \\
 \{y = 1 \wedge I\} & \{z = 1 \wedge I\} \\
 \{y = 1\} & \{z = 1\} \\
 & \text{coend} \\
 \{y = 1 \wedge z = 1 \wedge I\} \\
 \{x = 2\}
 \end{array}
 \end{array}$$

Owicki-Gries Proof Rules

$$\frac{r(\bar{x}) : I \vdash \{P_1\} \ c_1 \ \{Q_1\} \quad r(\bar{x}) : I \vdash \{P_2\} \ c_2 \ \{Q_2\}}{\{P_1 \wedge P_2 \wedge I\} \ \mathbf{resource} \ r(\bar{x}) : \ \mathbf{cobegin} \ c_1 \ // \ c_2 \ \mathbf{coend} \ \{Q_1 \wedge Q_2 \wedge I\}}$$

$$\frac{\{P \wedge I\} \ c \ \{Q \wedge I\}}{r(\bar{x}) : I \vdash \{P\} \ \mathbf{with} \ r \ \mathbf{do} \ c \ \{Q\}}$$

Owicki-Gries Proof Rules, with Conditions on Variable Occurrences

$$\frac{\begin{array}{l} \bar{y}; \bar{y}'; \bar{y}''; r(\bar{x}) : I \vdash \{P_1\} \ c_1 \ \{Q_1\} \\ \bar{z}; \bar{z}'; \bar{z}''; r(\bar{x}) : I \vdash \{P_2\} \ c_2 \ \{Q_2\} \\ \bar{y} \cap \bar{z}'\bar{z}'' = \emptyset \quad \bar{z} \cap \bar{y}'\bar{y}'' = \emptyset \quad \bar{y}'\bar{z}' \cap \bar{x} = \emptyset \quad \text{FV}(I) \subseteq \bar{x} \end{array}}{\{P_1 \wedge P_2 \wedge I\} \ \mathbf{resource} \ r(\bar{x}) : \mathbf{cobegin} \ c_1 \ // \ c_2 \ \mathbf{coend} \ \{Q_1 \wedge Q_2 \wedge I\}}$$

$$\frac{\bar{y}\bar{x}; \bar{y}'' \vdash \{P \wedge I\} \ c \ \{Q \wedge I\}}{\bar{y}; \bar{y}'; \bar{y}''; r(\bar{x}) : I \vdash \{P\} \ \mathbf{with} \ r \ \mathbf{do} \ c \ \{Q\}}$$

Contents

- 1 Refresher: Resource Invariants and Auxiliary Variables
 - Resource Invariants à la Hoare
 - Resource Invariants and Auxiliary Variables à la Owicki-Gries
- 2 The Procedure-Modularity Problem
- 3 Our Solution
- 4 Final Words
 - Dynamic Example: A Queue
 - More Goodies
 - Conclusion

The Example Program, Modularized

External synchronization

```
procedure inc() do  $x := x + 1$   
resource  $r(x)$  : cobegin  
    with  $r$  do inc()  
    //  
    with  $r$  do inc()  
coend
```

Internal synchronization

```
procedure inc(r) do with  $r$  do  $x := x + 1$   
resource  $r(x)$  : cobegin  
    inc(r)  
    //  
    inc(r)  
coend
```

External Synchronization: Proof of Complete Postcondition

$\{x = X\} \text{inc}() \{x = X + 1\}$

$\{x = 0\}$

$y := 0; z := 0;$

$\{y = 0 \wedge z = 0 \wedge I\}$

resource $r(x, y, z) : \text{cobegin} \quad I \equiv x = y + z$

$\{y = 0\}$

with r **do**

$\{y = 0 \wedge I\}$

begin $\text{inc}(); y := 1$ **end**

$\{y = 1 \wedge I\}$

$\{y = 1\}$

//

$\{z = 0\}$

with r **do**

$\{z = 0 \wedge I\}$

begin $\text{inc}(); z := 1$ **end**

$\{z = 1 \wedge I\}$

$\{z = 1\}$

coend

$\{y = 1 \wedge z = 1 \wedge I\}$

$\{x = 2\}$

The Example Program, Modularized

External synchronization

```
procedure inc() do  $x := x + 1$   
resource  $r(x)$  : cobegin  
    with  $r$  do inc()  
    //  
    with  $r$  do inc()  
coend
```

Internal synchronization

```
procedure inc(r) do with  $r$  do  $x := x + 1$   
resource  $r(x)$  : cobegin  
    inc(r)  
    //  
    inc(r)  
coend
```

Contents

- 1 Refresher: Resource Invariants and Auxiliary Variables
 - Resource Invariants à la Hoare
 - Resource Invariants and Auxiliary Variables à la Owicki-Gries
- 2 The Procedure-Modularity Problem
- 3 Our Solution**
- 4 Final Words
 - Dynamic Example: A Queue
 - More Goodies
 - Conclusion

Proposed Solution: Passing Auxiliary Variable Updates as an Argument

```
procedure inc(r, ρ) do  
  with r do begin x := x + 1; ρ end  
  
  y := 0; z := 0;  
resource r(x, y, z) : cobegin  
  inc(r, y := 1)  
  //  
  inc(r, z := 1)  
coend
```

Specification of Inc

$$\frac{
 \begin{array}{l}
 P \wedge I \Rightarrow U(x + 1) \\
 x \in \bar{y}'' \quad x \notin FV(U) \quad \bar{y}\bar{x}; \bar{y}'' \vdash \{U(x)\} \rho \{Q \wedge I\}
 \end{array}
 }{
 \bar{y}; \bar{y}'; \bar{y}''; r(\bar{x}) : I \vdash \{P\} \text{inc}(r, \rho) \{Q\}
 }$$

Proof of Inc

$$\frac{x \in \bar{y}'' \quad x \notin FV(U) \quad P \wedge I \Rightarrow U(x+1) \quad \bar{y}x; \bar{y}'' \vdash \{U(x)\} \rho \{Q \wedge I\}}{\bar{y}; \bar{y}'; \bar{y}''; r(\bar{x}) : I \vdash \{P\} \text{ inc}(r, \rho) \{Q\}}$$

$\{P\}$

with r **do begin**

$\{P \wedge I\}$

By premise 1

$\{U(x+1)\}$

$x := x + 1;$ *By premises 2,3*

$\{U(x)\}$

ρ *By premise 4*

$\{Q \wedge I\}$

end

$\{Q\}$

Proof of Client Program

```
{x = 0}
y := 0; z := 0;
{y = 0 ∧ z = 0 ∧ I}
resource r(x, y, z) : cobegin with I ≡ x = y + z
  {y = 0}
  inc(r, y := 1)
  with P ≡ y = 0; Q ≡ y = 1; U ≡ λX. X = 1 + z
  {y = 1}
//
  {z = 0}
  inc(r, z := 1)
  with P ≡ z = 0; Q ≡ z = 1; U ≡ λX. X = y + 1
  {z = 1}
coend
{y = 1 ∧ z = 1 ∧ I}
{x = 2}
```

Proof of Client Program

$$\frac{
 \begin{array}{c}
 P \wedge I \Rightarrow U(x+1) \\
 x \in \bar{y}'' \quad x \notin FV(U) \quad \bar{y}x; \bar{y}'' \vdash \{U(x)\} \quad \rho \quad \{Q \wedge I\}
 \end{array}
 }{
 \bar{y}; \bar{y}'; \bar{y}''; r(\bar{x}) : I \vdash \{P\} \text{inc}(r, \rho) \{Q\}
 }$$

$\{y = 0\}$

$\text{inc}(r, y := 1)$

with $P \equiv y = 0; Q \equiv y = 1; U \equiv \lambda X. X = 1 + z$

$\{y = 1\}$

$y = 0 \wedge x = y + z \Rightarrow x + 1 = 1 + z \quad x \in x, y \quad x \notin z$

$x, y, z; x, y \vdash \{x = 1 + z\} \quad y := 1 \quad \{y = 1 \wedge x = y + z\}$

$y; ; x, y; r(x, y, z) : x = y + z \vdash \{y = 0\} \text{inc}(r, y := 1) \{y = 1\}$

Contents

- 1 Refresher: Resource Invariants and Auxiliary Variables
 - Resource Invariants à la Hoare
 - Resource Invariants and Auxiliary Variables à la Owicki-Gries
- 2 The Procedure-Modularity Problem
- 3 Our Solution
- 4 Final Words
 - Dynamic Example: A Queue
 - More Goodies
 - Conclusion

Dynamic Example: A Queue

- Sequential spec:

$$\{\text{queue}(o, \alpha)\} \text{ enqueue}(o, v) \{\text{queue}(o, \alpha \cdot v)\}$$

- Proposed fine-grained spec:

$$\frac{I_A(s) * S \Leftrightarrow \exists \alpha \bullet \text{queue}(o, \alpha) * R(\alpha) \quad \forall \alpha \bullet \{R(\alpha) * P\} \rho \{R(\alpha \cdot v) * Q\}}{\{\pi A(s) * S * P\} \text{ enqueue}(s, o, v, \rho) \{\pi A(s) * S * Q\}}$$

where

- s atomic space that protects the queue
- $\pi A(s)$ permission to access atomic space s
- $I_A(s)$ invariant of atomic space s
- ρ auxiliary variable updates to re-establish $I_A(s)$

Dynamic Example: A Queue with Ownership Transfer

- Sequential spec:

$$\{\text{queue}(o, \alpha) * \text{node}(n, v)\} \text{enqueue}(o, n) \{\text{queue}(o, \alpha \cdot v)\}$$

- Proposed fine-grained spec:

$$I_A(s) * S \Leftrightarrow \exists \alpha \bullet \text{queue}(o, \alpha) * R(\alpha) \\ \forall \alpha \bullet \{R(\alpha) * P\} \rho \{R(\alpha \cdot v) * Q\}$$

$$\{\pi A(s) * S * P * \text{node}(n, v)\} \text{enqueue}(s, o, n, \rho) \{\pi A(s) * S * Q\}$$

where

- s atomic space that protects the queue
- $\pi A(s)$ permission to access atomic space s
- $I_A(s)$ invariant of atomic space s
- ρ auxiliary variable updates to re-establish $I_A(s)$

More Goodies

- in the paper:
 - Elaboration for dynamic locks and dynamic data structures, using separation logic
 - Formalization and soundness proof
 - Example: concurrent set as lock-coupled sorted list
- on the web:
 - Coq proof
 - Implementation in VeriFast verifier for C
 - Verified C implementations of lock-coupling set, multiple-compare-and-swap (MCAS)

Conclusion

We propose an approach
for
specifying and verifying fine-grained concurrent modules
that is
more powerful* and simpler*
but not necessarily easier to use
than
the best existing approaches
which are
Cave
(a combination of linearizability, rely-guarantee, and separation logic)
and
Concurrent Abstract Predicates