Owicki-Gries Logic

Sources


Concurrency

The Hoare Logic we have studied so far only considers sequential programs. How does concurrency affect the framework?

To study this question we will:

- add a construct to the programming language for spawning threads, and
- study the impact this has on the proof rules for Hoare triples.

The new logic is often called *Owicki-Gries Logic*, after the researchers who developed the idea.
The `cobegin` Statement

... introduces concurrency into GC. It has following form.

\[
\text{cobegin } S_1 \parallel S_2 \parallel \ldots \parallel S_n \text{ coend}
\]

- Each \( S_i \) can have a label, \( L_i \) (prepended to the statement using double colons ::).
- Control is concurrently enabled at each of the \( S_i \)'s; after all have terminated, the construct is exited and a single thread of control re-established.
- Shared memory!
- We will call the extended language \( PGC \) (Parallel GC).
Owicki-Gries Logic Verification Framework: Sys

Adding `cobegin... coend` to the programming notation alters `Sys`, so we must reopen the issue of program semantics.
Subtleties in the Semantics of \( \text{cobegin} \)

In sequential programming the following are equivalent.

\[
\begin{align*}
y &:= x; \\
x, y &:= x+1, x \\
x &:= y+1 \\
S_1 &
\end{align*}
\]

\[
\begin{align*}
y &:= y+1 \\
S_2 &
\end{align*}
\]

(Here “equivalent” means \([S_1] = [S_2]\).) But what happens if they are substituted for \(S\) in

\[
\text{cobegin } S \parallel y := y+1 \text{ coend}
\]

**Moral** In concurrent programming, issues of *atomicity* and *nondeterminism* must be factored into semantics.

\( \Rightarrow \) Existing semantics of programming language must be changed!
Fix $S$ to be the set of PGC statements (possibly containing occurrences of `cobegin`).

The new semantics will be given in an SOS transition ("small-step") style, as a relation $\rightarrow \subseteq (S \times \Sigma) \times (\Sigma \cup (S \times \Sigma))$.

**Intuition** $\rightarrow$ captures a notion of "atomic execution step":

- $\langle S, \sigma \rangle \rightarrow \langle S', \sigma' \rangle$ if in state $\sigma$, $S$ can engage in one execution step and then behave like $S'$, with the state changing from $\sigma$ to $\sigma'$.
- $\langle S, \sigma \rangle \rightarrow \sigma'$ if $S$ can terminate in one step, with the state changing to $\sigma'$.

How do we define $\rightarrow$? Using inference rules, just as we did for the evaluation semantics of PC.
Rules for $\rightarrow$

$$\langle \text{skip}, \sigma \rangle \rightarrow \sigma$$

$$k_1 = \lfloor e_1 \rfloor_{AE}(\sigma) \quad \ldots \quad k_n = \lfloor e_n \rfloor_{AE}(\sigma)$$

$$\langle v_1, \ldots, v_n := e_1, \ldots e_n, \sigma \rangle \rightarrow \sigma[v_1 \mapsto k_1, \ldots, v_n \mapsto k_n]$$

$$\langle S_1, \sigma \rangle \rightarrow \sigma'$$

$$\langle S_1; S_2, \sigma \rangle \rightarrow \langle S_2, \sigma' \rangle$$

$$\langle S_1, \sigma \rangle \rightarrow \langle S_1', \sigma' \rangle$$

$$\langle S_1; S_2, \sigma \rangle \rightarrow \langle S_1'; S_2, \sigma' \rangle$$
Rules for $\rightarrow$ (cont.)

\[
\begin{align*}
\sigma &\models_{BE} G_i \\
\langle \text{if } G_1 \to S_1 \cdots \cdots G_n \to S_n \; \text{fi}, \sigma \rangle &\rightarrow \langle S_i, \sigma \rangle
\end{align*}
\]

\[
\begin{align*}
\sigma &\not\models_{BE} G_1 \cdots \sigma \not\models_{BE} G_n \\
\langle \text{do } G_1 \to S_1 \cdots \cdots G_n \to S_n \; \text{od}, \sigma \rangle &\rightarrow \sigma
\end{align*}
\]

\[
\begin{align*}
\sigma &\models_{BE} G_i \\
\langle \text{do } G_1 \to S_1 \cdots \cdots G_n \to S_n \; \text{od}, \sigma \rangle &\rightarrow \langle S_i; \; \text{do } G_1 \to S_1 \cdots \cdots G_n \to S_n \; \text{od}, \sigma \rangle
\end{align*}
\]
Rules for $\rightarrow$ (cont.)

\[ \langle S, \sigma \rangle \rightarrow \sigma' \]

\[ \langle \text{cobegin } S \text{ coend}, \sigma \rangle \rightarrow \sigma' \]

\[ n \geq 2 \quad \langle S_i, \sigma \rangle \rightarrow \sigma' \]

\[ \langle \text{cobegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel S_i \parallel S_{i+1} \parallel \cdots \parallel S_n \text{ coend}, \sigma \rangle \rightarrow \]
\[ \langle \text{cobegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel S_{i+1} \parallel \cdots \parallel S_n \text{ coend}, \sigma' \rangle \]

\[ \langle S_i, \sigma \rangle \rightarrow \langle S'_i, \sigma' \rangle \]

\[ \langle \text{cobegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel S_i \parallel S_{i+1} \parallel \cdots \parallel S_n \text{ coend}, \sigma \rangle \rightarrow \]
\[ \langle \text{cobegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel S'_i \parallel S_{i+1} \parallel \cdots \parallel S_n \text{ coend}, \sigma' \rangle \]

**Note** This semantics models concurrency as *interleaving* of atomic execution steps in the individual threads.
Owicki-Gries Logic: Spec and sat

The set $\textit{Spec}$ of specifications will remain unchanged: precondition/postcondition pairs. What about $\textit{sat} \subseteq \textit{Sys} \times \textit{Spec}$?

**Definition**  Program $S$ satisfies specification $\langle P, Q \rangle$ if for every $\sigma, \sigma' \in \Sigma$ such that $\sigma \models P$ and $\langle S, \sigma \rangle \rightarrow^* \sigma'$, $\sigma' \models Q$.

**Aside (Transitive and Reflexive Closure)**  What is $\rightarrow^*$?

- Let $R \subseteq T \times U$ be a binary relation, with $T \subseteq U$.
- Then $R^* \subseteq T \times U$, the reflexive and transitive closure of $R$, defined as follows.
  - $t R^* t$ for all $t \in T$.
  - If $t R t'$ and $t' R^* u$, then $t R^* u$.

Idea: $t R^* u$ holds if $t$ can reach $u$ via a “sequence” of $R$ steps: $t \equiv t_1 R \cdots R t_n \equiv u$.

**Aside (Full Abstraction)**  What is the relationship between $\Rightarrow$ and $\rightarrow$? Suppose $S$ is a GC program (i.e. no $\text{cobegin}$). Then for any $\sigma, \sigma' \in \Sigma$:

$\langle S, \sigma \rangle \Rightarrow \sigma'$ iff $\langle S, \sigma \rangle \rightarrow^* \sigma'$
Because of full abstraction, all the old Hoare Logic inference rules remain valid!

We need to add a rule for \texttt{cobegin}...\texttt{coend}. The following is due to Owicki and Gries.

\[
\begin{aligned}
\{P_1\} S_1 \{Q_1\} \cdots \{P_n\} S_n \{Q_n\} & \quad \text{interference freedom} \\
\{P_1 \land \cdots \land P_n\} \texttt{cobegin} S_1 \parallel \cdots \parallel S_n \texttt{coend} \{Q_1 \land \cdots \land Q_n\} & \quad \text{(cobegin)}
\end{aligned}
\]
... a property of the proofs of the $\{P_i\} S_i \{Q_i\}$

- Suppose we have a proof for $\{P_i\} S_i \{Q_i\}$.

- Interference-freedom requires proving that the execution of statement in another $S_j$ does not invalidate the reasoning used in the proof of $\{P_i\} S_i \{Q_i\}$.
The conclusion $x = 1$ is bogus!

But the individual reasoning of each parallel statement is fine!

Executing one assignment to $x$ invalidates all the state predicates ($x = 0, x + 1 = 1$) preceding the other! This phenomenon is called interference; correct proofs must avoid it.
Formalizing Interference-Freedom

... turns out to be somewhat tricky.

- Interference-freedom is a property of proofs, not Hoare triples.
- Identifying which parts of a proof need to be considered requires some effort.

To cope with these issues, we will do the following.

- Define the notion of normalized proof outline, which is like a proof outline but with only one predicate before every statement and one predicate at end.
- Define interference-freedom using normalized proof outlines.
Normalized Proof Outlines

... like proof outlines, but with exactly one state predicate before each statement and at end.

Example

\{
\tt
\}
\begin{align*}
x &:= 0; \\
\{x = 0\}
\end{align*}
\begin{align*}
\text{cobegin} \\
\{x = 0\} \{x + 1 = 1\} \\
x &:= x + 1 \\
\{x = 1\}
\end{align*}
\begin{align*}
\parallel \{x = 0\} \{x + 1 = 1\} \\
x &:= x + 1 \\
\{x = 1\}
\end{align*}
\begin{align*}
\text{coend} \\
\{x = 1\}
\end{align*}
**Facts about Normalized Proof Outlines**

**Definition**
Let $N_S$ be a normalized proof outline for statement $S$.

1. If $S'$ is a statement in $S$, then let $\text{pre}(N_S, S')$ and $\text{post}(N_S, S')$ be the state predicates immediately preceding and following $S'$, respectively. Write $\text{pre}(N_S)$ for $\text{pre}(N_S, S)$ and $\text{post}(N_S)$ for $\text{post}(N_S, S)$.

2. $N_S$ is valid if for every statement $S'$ in $S$, the Hoare triple $\{\text{pre}(N_S, S')\} S' \{\text{post}(N_S, S')\}$ is valid.

**Fact**

1. Let $S$ be a cobegin-free program. Then $\{P\} S \{Q\}$ is valid if and only if there is a valid normalized proof outline $N_S$ for $S$ with $\text{pre}(N_S) = P$ and $\text{post}(N_S) = Q$.

2. Every valid normalized proof outline may be converted into a (full) proof outline.
Interference-Freedom and Normalized Proof Outlines

Recall the \((\text{cobegin})\) rule:

\[
\frac{\{P_1\} S_1 \{Q_1\} \cdots \{P_n\} S_n \{Q_n\}}{\{P_1 \wedge \cdots \wedge P_n\} \text{cobegin } S_1 \parallel \cdots \parallel S_n\text{ coend } \{Q_1 \wedge \cdots \wedge Q_n\}} \quad \text{(cobegin)}
\]

- Suppose each \(S_i\) has a normalized proof outline \(N_{S_i}\).

- \(N_{S_i}\) is \textit{interference-free} with respect to proof outline \(N_{S_j}\) \((i \neq j)\) if for each statement \(S'_i\) in \(S_i\) and \(S'_j\) in \(S_j\):

\[
\{\text{pre}(N_{S_i}, S'_i) \wedge \text{pre}(N_{S_j}, S'_j)\} S'_i \{\text{pre}(N_{S_i}, S'_i)\}
\]

\[
\{\text{post}(N_{S_i}, S'_i) \wedge \text{pre}(N_{S_j}, S'_j)\} S'_j \{\text{post}(N_{S_i}, S'_i)\}
\]

The \(N_{S_1}, \ldots, N_{S_n}\) are interference free if they are pairwise interference free with respect to one other.

- So applying the \((\text{cobegin})\) rule requires the development of interference-free normalized proof outlines for the \(S_i\)!

- In proving interference-freedom of \(N_{S_i}\) with respect to \(N_{S_j}\), can limit our attention to:
  - Preconditions of each statement in \(S_i\), and postcodition of \(N_{S_i}\)
  - Assignment statements in \(S_j\)
Example Revisited

\[
\{ \text{tt} \} \\
x := 0; \\
\{ x = 0 \} \\
\text{cobegin} \\
\{ x = 0 \} \\
x := x + 1 \\
\{ x = 1 \} \\
\parallel \{ x = 0 \} \\
x := x + 1 \\
\{ x = 1 \} \\
\text{coend} \\
\{ x = 1 \}
\]

Interference-freedom is violated because the following Hoare triple is not valid:

\[
\{ x = 0 \land x = 0 \} \ x := x + 1 \ \{ x = 0 \}
\]
Another Example

We want to prove:

\[
\{ \text{bal} = B \land \text{amt} > 0 \} \\
S \\
\{ \text{bal} = B + \text{amt} \land \text{amt} > 0 \land (\text{credit} = 1 \Rightarrow \text{bal} > 1000) \}
\]

where \( S \) is given below.

\[
\begin{align*}
\text{cobegin} \\
& \quad S_{\text{amt}} :: \quad \text{bal} := \text{bal} + \text{amt} \\
\parallel & \quad S_{\text{cred}} :: \quad \text{if} \\
& \quad \quad \quad \text{bal} > 1000 \rightarrow \text{credit} := 1 \\
& \quad \quad \quad \text{bal} \leq 1000 \rightarrow \text{credit} := 0 \\
\text{fi} \\
\text{coend}
\end{align*}
\]
Proof of Example

We do the proof as follows.

1. Give valid normalized proof outline for
   \[ \{ \text{bal} = B \land \text{amt} > 0 \} \text{S}_{\text{amt}} \{ \text{bal} = B + \text{amt} \land \text{amt} > 0 \} \]

2. Give valid normalized proof outline for
   \[ \{ \text{tt} \} \text{S}_{\text{cred}} \{ \text{credit} = 1 \Rightarrow \text{bal} > 1000 \} \]

3. Use \texttt{cobegin} rule to infer desired result.

   Outline for \text{S}_{\text{amt}}

   \[ \{ \text{bal} = B \land \text{amt} > 0 \} \]
   \[ \text{bal} \ := \ \text{bal} \ + \ \text{amt} \]
   \[ \{ \text{bal} = B + \text{amt} \land \text{amt} > 0 \} \]

©2015 Rance Cleaveland. All rights reserved.
Outline for $S_{\text{cred}}$

{tt}
if
  bal > 1000 →
  {bal > 1000}
  credit := 1
  {credit = 1 ⇒ bal > 1000}
  fi
  bal ≤ 1000 →
  {tt}
  credit := 0
  {credit = 1 ⇒ bal > 1000}
fi
{credit = 1 ⇒ bal > 1000}
Applying \texttt{cobegin}

To apply \texttt{(cobegin)} we need to prove \textit{non-interference}. For convenience, define the following.

\begin{align*}
\mathcal{D}_1 &= (\text{bal} = B) \land (\text{amt} > 0) \\
\mathcal{D}_2 &= (\text{bal} = B + \text{amt}) \land (\text{amt} > 0) \\
\mathcal{C}_1 &= \text{tt} \\
\mathcal{C}_2 &= \text{bal} > 1000 \\
\mathcal{C}_3 &= \text{credit} = 1 \Rightarrow \text{bal} > 1000
\end{align*}

\(\mathcal{D}_1, \mathcal{D}_2\) are state predicates in outline for \(S_{\text{amt}}\); \(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\) are those for \(S_{\text{cred}}\).

To prove non-interference we must establish, for all \(1 \leq i \leq 2\) and \(1 \leq j \leq 3\):

1. \(\{C_j \land D_1\} \text{ bal} := \text{bal} + \text{amt} \{C_j\}\)
2. \(\{D_i \land C_2\} \text{ credit} := 1 \{D_i\}\)
3. \(\{D_i \land C_1\} \text{ credit} := 0 \{D_i\}\)

7 proof obligations!
Most are vacuously true, however.

- Triples of types (2) and (3) hold because no $D_i$ mentions variable credit.
- Assertion $C_1$ is true in all states and hence cannot be invalidated.

Consequently, to complete the proof of non-interference we must prove:

- $\{C_2 \land D_1\} \text{bal} := \text{bal} + \text{amt} \{C_2\}$
- $\{C_3 \land D_1\} \text{bal} := \text{bal} + \text{amt} \{C_3\}$
Proving Interference-Freedom: \( \{C_2 \land D_1\} \) \( \text{bal} := \text{bal} + \text{amt} \) \( \{C_2\} \)

\( \{(\text{bal} > 1000) \land (\text{bal} = B) \land (\text{amt} > 0)\} \)
\( \{\text{bal} > 1000 \land \text{amt} > 0\} \)
\( \{\text{bal} + \text{amt} > 1000\} \)
\( \text{bal} := \text{bal} + \text{amt} \)
\( \{\text{bal} > 1000\} \)
Proving Interference-Freedom: \( \{C_3 \land D_1\} \, \text{bal} := \text{bal} + \text{amt} \, \{C_3\} \)

\[
\{(\text{credit} = 1 \Rightarrow \text{bal} > 1000) \land (\text{bal} = B) \land (\text{amt} > 0)\} \\
\{\text{credit} = 1 \Rightarrow \text{bal} + \text{amt} > 1000\} \\
\text{bal} := \text{bal} + \text{amt} \\
\{\text{credit} = 1 \Rightarrow \text{bal} > 1000\} \\
\]

QED!
• If $S_{amt}$ had been a withdrawal transaction, the assignment statement would have been

$$bal := bal - amt$$

and the last step of the non-interference proof would not have gone through.

• Postcondition does \textit{not} say that credit denial implies a balance of 1000 or less, so a program that never grants credit would satisfy the given specification! I.e. we could change $S_{cred}$ to

$$credit := 0$$

and still conduct a proof of the same Hoare triple.

• We would like for a postcondition of the form

$$\left((credit = 1 \Rightarrow bal > 1000) \land (credit = 0 \Rightarrow bal \leq 1000)\right)$$

But this would lead to a violation of interference freedom. Why?
Soundness and Completeness

Theorem (Soundness) If \(\{P\} S \{Q\}\) is provable using the proof rules seen so far then \(S\) satisfies specification \( \langle P, Q \rangle \).

What about completeness?

- Completeness does not hold.
- Neither does *relative completeness*!
Incompleteness

**Theorem**  The following valid Hoare triple cannot be proved using the rules given so far.

\[
\{\text{tt}\} \quad \text{cobegin} \quad x := x + 2 \quad \parallel \quad x := 0 \quad \text{coend} \quad \{x = 0 \lor x = 2\}
\]

**Proof**  By contradiction. Suppose there were such a proof. Then there would be \(Q, R\) such that

\[
\{\text{tt}\} \quad x := x + 2 \quad \{Q\} \\
\{\text{tt}\} \quad x := 0 \quad \{R\} \\
Q \land R \quad \Rightarrow \quad x = 0 \lor x = 2
\]

By \((:=)\), \(tt \Rightarrow Q^x_{x+2}\) holds, and hence \(Q\) does also. Similarly, \(R^x_0\) holds. By \((\text{cobegin})\), \(\{R \land \text{tt}\} \quad x := x + 2 \quad \{R\}\) holds, meaning \(R \Rightarrow R^x_{x+2}\) is valid. But then by induction, \(\forall x. (x \geq 0 \land \text{even}(x)) \Rightarrow R\) is true. Since \(Q \land R \Rightarrow x = 0 \lor x = 2\), it now follow that \(\forall x. (x \geq 0 \land \text{even}(x)) \Rightarrow (x = 0 \lor x = 2)\), which is a contradiction: there are positive even integers other than 0 or 2.
We have just showed that for every interference-free proof outline of form:

\[
\begin{align*}
\{tt\} & \\
cobegin & \\
\{tt\} x := x + 2 \{Q\} & \\
\| & \\
\{tt\} x := 0 \{R\} & \\
coend & \\
\{Q \land R\} & \\
\end{align*}
\]

\(R\) must hold for all even, positive \(x\).

- \(R\) must hold after execution of \(x := 0\).
- \(R\) must also hold both before and after execution of \(x := x + 2\).

What is needed is the capability in \(R\) to say that \textit{until} \(x := x + 2\) fires, \(x = 0\) holds. This can be done using \textit{auxiliary variables}.
Auxiliary Variables

... variables that are put into a program just to reason about progress in other processes.

done := 0;
cobegin
  x, done := x+2, 1
∥  x := 0
coen

• done is auxiliary: it is only assigned to, and not read

• It is 0 when increment to x is pending and 1 when it is completed.

• Proof is now possible!
Normalized Proof Outline Using Auxiliary Variables

\{tt\}
done := 0;
\{done = 0\}
cobegin
  \{done = 0\}
x, done := x+2, 1
  \{tt\}
||
  \{tt\}
x := 0
  \{(x = 0 \lor x = 2) \land (\text{done} = 0 \Rightarrow x = 0)\}
coend
\{x = 0 \lor x = 2\}
Non-interference Proofs

- \{\text{done} = 0 \land \text{tt}\} \ x := 0 \ \{\text{done} = 0\}
- \{\text{tt} \land \text{tt}\} \ x := 0 \ \{\text{tt}\}
- \{\text{tt} \land \text{done} = 0\} \ x, \text{done} := x + 2, 1 \ \{\text{tt}\}
- \{(x = 0 \lor x = 2) \land (\text{done} = 0 \Rightarrow x = 0) \land \text{done} = 0\}
  \{x = 0\}
  \{x + 2 = 2 \land 1 = 1\}
  x, \text{done} := x+2, 1
  \{x = 2 \land \text{done} = 1\}
  \{(x = 0 \lor x = 2) \land (\text{done} = 0 \Rightarrow x = 0)\}
... adding auxiliary variables enables proofs to be conducted

... but we don’t want these variables to be in our code!

Fortunately, auxiliary variables can be removed.

\[
\frac{\{P\} S \{Q\} \quad x \text{ not free in } Q \quad x \text{ auxiliary in } S}{\{P\} S' \{Q\}} \quad (\text{aux})
\]

where \( S' \) is \( S \) with all references to \( x \) deleted.

Using (aux), references to done can be removed!

Theorem (Relative Completeness) Adding rules (cobegin) and (aux) to the other Hoare rules yields a relatively complete proof system for the cobegin language.