CSCI 740 - Programming Language Theory

Lecture 26

Hoare Logic for Concurrent Programs

Instructor: Hossein Hojjat

November 3, 2017

- Is the following true?

```
{x = 0}
y := x;
x := x + 1;
{x = 1 ∧ y = 0}
```

- YES!

Program Verification with Hoare Triples

- Is the following still true?

```
{x = 0}
y := x;                 ||        x := 5;
x := x + 1;
{x = 1 ∧ y = 0}
```

Program Verification with Hoare Triples

- Is the following still true?

$\{x = 0\}$
`y := x;`
`x := x + 1;`
$\{x = 1 \wedge y = 0\}$
	$\|\|$	`x := 5;`

- NO!

Program Verification with Hoare Triples

- Is the following still true?

$$
\begin{array}{lcl}
\{x = 0\} & & \\
\texttt{y := x;} & \quad\|\quad & \texttt{x := 5;} \\
\{x + 1 = 1 \land y = 0\} & & \\
\texttt{x := x + 1;} & & \\
\{x = 1 \land y = 0\} & &
\end{array}
$$

- NO!

Program Verification with Hoare Triples

- Is the following still true?

```
{x = 0}
y := x;
~~{x + 1 = 1 ∧ y = 0}~~
x := x + 1;
{x = 1 ∧ y = 0}
```

||

```
x := 5;
```

- NO!
- The parallel process may interfere with the intermediate assertions

Parallel Composition

- Extend IMP language of previous lectures with parallel composition

$$
\begin{aligned}
e ::= \;& n \mid x \mid e_1 + e_2 \mid e_1 = e_2 \\
c ::= \;& x := e \mid \texttt{if}\; e\; \texttt{then}\; c_1 \;\texttt{else}\; c_2 \mid \\
& \texttt{while}\; e\; \texttt{do}\; c \mid \texttt{skip} \mid c_1 \,;\, c_2 \mid \\
& c_1 \parallel c_2
\end{aligned}
$$

Rule for Parallel Composition

- Can we derive a Hoare triple for parallel composition from the triples of each command?

First Attempt:

$$\frac{\vdash \{P_1\}\ c_1\ \{Q_1\} \qquad \vdash \{P_2\}\ c_2\ \{Q_2\}}{\vdash \{P_1 \land P_2\}\ c_1 \parallel c_2\ \{Q_1 \land Q_2\}}$$

- **Intuition:** if we satisfy the preconditions of c_1 and c_2, their postconditions will be satisfied too

Unsoundness of First Attempt

$$\frac{\vdash \ \{P_1\}\ c_1\ \{Q_1\} \qquad \vdash \ \{P_2\}\ c_2\ \{Q_2\}}{\vdash \ \{P_1 \land P_2\}\ c_1 \parallel c_2\ \{Q_1 \land Q_2\}}$$

- This rule is not always sound, consider:

$\{x = 1\}\ \ y := 0\ \ \{x = 1\}$ $\qquad$ $\{\textit{true}\}\ \ x := 10\ \ \{\textit{true}\}$

- It does not hold that

$$\{x = 1 \land \textit{true}\}\ \ y := 0 \parallel x := 10\ \ \{x = 1 \land \textit{true}\}$$

Second Attempt

$$\frac{\vdash \ \{P_1\}\ c_1\ \{Q_1\} \qquad \vdash \ \{P_2\}\ c_2\ \{Q_2\}}{\vdash \ \{P_1 \wedge P_2\}\ c_1 \parallel c_2\ \{Q_1 \wedge Q_2\}}$$

- If c_1 and c_2 do not read and write the same variables, and all the pres- and post- conditions talk about different variables
- What's wrong with this?

Second Attempt

$$\frac{\vdash \ \{P_1\}\ c_1\ \{Q_1\} \qquad \vdash \ \{P_2\}\ c_2\ \{Q_2\}}{\vdash \ \{P_1 \land P_2\}\ c_1 \parallel c_2\ \{Q_1 \land Q_2\}}$$

- If c_1 and c_2 do not read and write the same variables,
  and all the pres- and post- conditions talk about different variables
- What's wrong with this?
- No way to prove some program
- The rule is **incomplete**

Third Attempt

Let $\mathit{UPD}(c)$ be the set of variables that are updated (modified) in c

$$\frac{\vdash \ \{P_1\}\ c_1\ \{Q_1\} \qquad \vdash \ \{P_2\}\ c_2\ \{Q_2\}}{\vdash \ \{P_1 \wedge P_2\}\ c_1 \parallel c_2\ \{Q_1 \wedge Q_2\}}$$

If $\mathit{UPD}(c_1) \cap (\mathit{FV}(P_2) \cup \mathit{FV}(Q_2)) = \emptyset$ and $\mathit{UPD}(c_2) \cap (\mathit{FV}(P_1) \cup \mathit{FV}(Q_1)) = \emptyset$

Third Attempt

Let $\mathit{UPD}(c)$ be the set of variables that are updated (modified) in c

$$\frac{\vdash\ \{P_1\}\ c_1\ \{Q_1\} \qquad \vdash\ \{P_2\}\ c_2\ \{Q_2\}}{\vdash\ \{P_1 \wedge P_2\}\ c_1 \parallel c_2\ \{Q_1 \wedge Q_2\}}$$

If $\mathit{UPD}(c_1) \cap (\mathit{FV}(P_2) \cup \mathit{FV}(Q_2)) = \emptyset$ and $\mathit{UPD}(c_2) \cap (\mathit{FV}(P_1) \cup \mathit{FV}(Q_1)) = \emptyset$

Still unsound. Consider:

$\{x = 0\}\ \ y := x; z := y\ \ \{z = 0\}$ $\qquad\qquad$ $\{\mathit{true}\}\ \ y := 10\ \ \{\mathit{true}\}$

It does not hold that

$$\{x = 0 \wedge \mathit{true}\}\ \ y := x; z := y \parallel y := 10\ \ \{z = 0 \wedge \mathit{true}\}$$

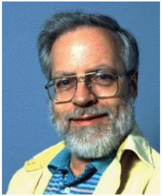
Diagnose: $y := 10$ interferes with the proof of

$$\{x = 0\}\ \ y := x; z := y\ \ \{z = 0\}$$

$$y = 0$$

Owicki-Gries Reasoning

- Susan Owicki,
  "Axiomatic proof techniques for parallel programs",
  Cornell University, Ithaca, NY, 1975
  - Under supervision of Prof. David Gries
- First complete logic for partial correctness of concurrent programs that communicate using shared variables



Interference Freedom

- **Interference Freedom:** every assertion used in the local verification is not invalidated by the execution of the other process

P_1:	P_2:
$\{p_1\}$	$\{q_1\}$
c_1	a_1
$\{p_2\}$	$\{q_2\}$
c_2	a_2
$\ldots$	$\ldots$

We say that they are interference free iff

$$\boxed{\begin{array}{c} \forall p_i \in \text{assertions of } P_1 \wedge \forall a_j \in \text{atomic actions of } P_2, \\ \{p_i \wedge \text{pre } a_j\} \\ a_j \\ \{p_i\} \\ \text{(and vice versa)} \end{array}}$$

- If P_1 has n statements and P_2 has m statements, proving interference freedom requires proving $O(n \times m)$ correctness formulas

Example

- These two proof outlines are correct but not interference free
- For example, the assertion $x = 0$ is not preserved against the atomic action $x := x + 2$

$$\begin{array}{c} \{x = 0\} \\ x := x + 2; \\ \{x = 2\} \end{array} \quad \Big\| \quad \begin{array}{c} \{\mathit{true}\} \\ x := 0; \\ \{x = 0\} \end{array} \qquad \begin{array}{c} \{x = 0 \wedge x = 0\} \\ x := x + 2; \\ \{x = 0\} \end{array}$$

- By weakening the postconditions we obtain both correct and interference free proof outlines:

$$\begin{array}{c} \{x = 0\} \\ x := x + 2; \\ \{x = 0 \vee x = 2\} \end{array} \Big\| \begin{array}{c} \{\mathit{true}\} \\ x := 0; \\ \{x = 0 \vee x = 2\} \end{array} \quad \begin{array}{c} \{(x = 0 \vee x = 2) \wedge x = 0\} \\ x := x + 2; \\ \{x = 0 \vee x = 2\} \end{array} \quad \Rightarrow \quad \begin{array}{c} \{x = 0\} \\ x := x + 2; \\ \{x = 0 \vee x = 2\} \end{array}$$

Rule for Parallel Composition

$$\frac{\vdash \ \{P_1\}\ c_1\ \{Q_1\} \qquad \vdash \ \{P_2\}\ c_2\ \{Q_2\} \qquad \textit{interference freedom}}{\vdash \ \{P_1 \wedge P_2\}\ c_1 \parallel c_2\ \{Q_1 \wedge Q_2\}}$$

- This rule is **not** compositional
- A change in one of the components may affect the proof, not only of the modified component, but also of all the others

Rely-Guarantee

- **Rely-Guarantee** is a well-known compositional method for proving Hoare logic properties of concurrent programs
- Rough idea: instead of trying to write interference-free proofs, explicitly account for the allowed interference
- No additional interference checks required
- Pioneered by Cliff Jones (1981, 1983)

Rely-Guarantee

$$\boxed{R, G \vdash \{P\}\ c\ \{Q\}}$$

If

1. program c is executed in a state which satisfies P
2. every state change by another process satisfies R

then

1. every final state of c satisfies G
2. if the execution terminates, the final state will satisfy Q