Hoare logic (Part 2)

Grammar of the language

Integer expressions	$E ::= n \mid x \mid (-E) \mid (E + E)$
	$\mid (E - E) \mid (E * E)$
Boolean expressions	$B ::= \texttt{true} \mid \texttt{false} \mid (!B)$
	$\mid (B \& B) \mid (B \| B) \mid (E < E)$
	$\mid (E == E) \mid (E != E)$
Commands	$C ::= \quad x = E \mid C; C$
	$\mid \texttt{if}\ B\ \texttt{then}\ \{C\}\ \texttt{else}\{C\}$
	$\mid \texttt{while}\ B\ \{C\}$

Partial correctness vs. total correctness

Two semantic relations, two logical judgments:

	semantics	logic	name
	$\models_{par} (\|\phi\|) C (\|\psi\|)$	$\vdash_{par} (\|\phi\|) C (\|\psi\|)$	partial correctness
	$\models_{tot} (\|\phi\|) C (\|\psi\|)$	$\vdash_{tot} (\|\phi\|) C (\|\psi\|)$	total correctness

Weakest precondition for "if"

Suppose that we want the weakest precondition for

$$\texttt{if}\, B\, \texttt{then}\, \{C_1\}\, \texttt{else}\{C_2\},$$

given postcondition ψ. We proceed as follows:

1. Push ψ upward through C_1; call the result ϕ_1.
2. Push ψ upwards through C_2; call the result ϕ_2.
3. Set ϕ to be $(B \rightarrow \phi_1) \wedge (\neg B \rightarrow \phi_2)$.

Weakest precondition for "if"

Proposition. Let

1. $C = (\texttt{if}\, B\, \texttt{then}\, \{C_1\}\, \texttt{else} \{C_2\})$,
2. ϕ_1 resp. ϕ_2 be the weakest preconditions of $C_1(\llbracket\psi\rrbracket)$ resp. $C_2(\llbracket\psi\rrbracket)$, and
3. $\phi = (B \rightarrow \phi_1) \wedge (\neg B \rightarrow \phi_2)$.

Then ϕ is the weakest precondition of $C(\llbracket\psi\rrbracket)$.

Proof. That ϕ is a precondition: exercise (takes some work). Importantly, the proof works for both partial and total correctness.

Using the Partial-while rule

Recall the Partial-while rule:

$$\frac{(\!|\eta \wedge B|\!)\,C\,(\!|\eta|\!)}{(\!|\eta|\!)\,\texttt{while}\,B\,\{C\}\,(\!|\eta \wedge \neg B|\!)}\ \text{Partial-while}$$

What if we want to prove

$$\models_{par} (\!|\phi|\!)\,\texttt{while}\,B\,\{C\}\,(\!|\psi|\!)?$$

We must discover an η such that

- $\models \phi \rightarrow \eta$
- $\models \eta \wedge \neg B \rightarrow \psi$
- $\models_{par} (\!|\eta|\!)\,\texttt{while}\,B\,\{C\}\,(\!|\eta \wedge \neg B|\!)$.

Finding an invariant

- An η such that $\models_{par} (\!|\eta|\!)\,\mathtt{while}\, B\,\{C\}(\!|\eta \wedge \neg B|\!)$ is called an **invariant**.
- Finding an invariant requires ingenuity.
- For any while-statement there is more than one invariant, e.g. $\bot$ is an invariant for any loop, and so is $\top$.
- But most invariants (in particular $\bot$ and $\top$) are useless, because we also need

$$\vdash \phi \rightarrow \eta \quad \text{and} \quad \vdash \eta \rightarrow \psi.$$

Example program with while-loop

Recall the program Fac1:

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

We shall prove (in the lecture) that

$$\models_{par} (\!|\top|\!)\,\texttt{Fac1}\,(\!|y = x!|\!).$$

A calculus for total correctness

- The calculus presented so far proves only the **partial** correctness of triples, i.e. a proof of

$$(\!|\phi|\!)\,C\,(\!|\psi|\!)$$

  only talks about initial states that cause C to terminate.
- The only reason for non-termination are while-loops.
- So the calculus for **total** correctness differs from the one for partial correctness only in its treatment of `while`.

Total correctness of while-statements

A proof of total correctness for a while-statement consists of

- a proof of partial correctness, and
- a proof that the while-loop terminates.

Variants

- The proof of termination is given by an integer expression that decreases during each iteration, but remains non-negative.
- If such an expression exists, the loop terminates after finitely many iterations, because there are no infinite descending chains $n_0 > n_1 > n_2 > \ldots$ of non-negative integers.
- Such an integer expression is called a **variant**.

The Total-while rule

The Total-while rule is like the Partial-while rule, but with augmented pre- and postconditions:

$$\frac{(\!|\eta \land B \land (0 \leq E = E_0)|\!)\, C\, (\!|\eta \land (0 \leq E < E_0)|\!)}{(\!|\eta \land (0 \leq E)|\!)\, \texttt{while}\, B\, \{C\}\, (\!|\eta \land \neg B|\!)} \text{ Total-while.}$$

- E is the variant, which decreases during every iteration: if $E = E_0$ before the loop, then it is strictly less than E_0 after it—but it remains non-negative.
- Technically, E_0 is a variable that does not occur anywhere else.

Summary of Hoare logic (part 1/2)

- Hoare logic is for verifying properties of sequential, state-transforming programs.
- Hoare triples $(\!|\phi|\!)C(\!|\psi|\!)$ describe relationships between the states before and after running the program C.
- Hoare triples are either about total correctness or partial correctness, depending on whether C is required to terminate or not.

Summary of Hoare logic (part 2/2)

- Hoare logic is for proving Hoare triples.
- Hoare logic has a convenient tableaux method.
- Starting with the postcondition ψ, all proof steps for commands are mechanical, except for guessing invariants and variants of while-loops.
- Predicate logic is “imported” via the “Implied” rule.