Hoare Logic

Behavior of a statement given by a triple:

\[ \{P\} \; s \; \{Q\} \]

If \( P \) true of a state, and \( s \) executed, then \( Q \) true after

\( P \) and \( Q \) are logical properties of the state and effects

Need not (but can) exactly represent the state
Conditionals

In what cases is \{P\} if (e) \{ s \} else \{ t \} \{Q\} true?

- If e, same as \{P\} s \{Q\}
- If not e, same as \{P\} t \{Q\}

Triples already include the “if precondition” idea:

\{P \land e\} s \{Q\} \lor \{P \land \neg e\} t \{Q\}
Weakest Precondition

\[
\{P\} \ x = e \ \{Q\} \iff (P \rightarrow Q[x := e])
\]

\[
WP[x = e](Q) = Q[x := e]
\]

Common pattern: \( P \rightarrow \text{something}(Q) \)

Weakest precondition of \( Q \)

\[
\{P\} \ s \ \{Q\} \iff (P \rightarrow WP[s](Q))
\]
Class Progress

- Logical reasoning
- Program logics
- Static analysis

- Expressions
- Statements
- Loops
- Procedures
Loops

What are the **weakest preconditions of loops**?

- Imagining loops as infinitely-long sequences

**Loop invariants** for verification

- What does not change over a loop iteration

**Proving termination** using decreasing measures

- It can’t go below zero!
Verifying Loops

Loop invariants
Some loop examples

Which of these Hoare triples are true?

\[
\{ n = m \wedge i = 0 \} \\
\text{while } (n > 0) \{ i++; n--; \} \\
\{ i = m \} \\
\]

\[
\{ l \geq 0 \wedge n = 1 \} \\
k = 0 \\
\text{while } k < l: \\
k += 1 \\
n *= 2 \\
\{ n = 2^l \} \\
\]

\[
\{ l < r \} \\
\text{while } \text{rand}(): \\
t = l \\
l = r \\
r = t \\
\{ l < r \} \\
\]
What loops do

How do we describe the behavior of loops?

- **Conditional**: while \( e \) { \( s \) }
- **Only execute if true**: if \((e)\) { \( s \); while \( e \) { \( s \) } }
- **Recursive**: if \((e)\) { \( s \); if \((e)\) { \( s \); if \((e)\) { \( ... \) } } }

We can think of while loops as **infinite statements**

\[
\text{if (e) \{ s; if (e) \{ s; if (e) \{ ... \} \} \}}
\]
Hoare Triples

while e { s } = if (e) { s; while e { s } }

Use loop unrolling to convert Hoare triples to logic:

\{P\} while e { s } \{Q\}

\{P\} if e { s; while e { s } } \{Q\}

\((P \land \neg e \rightarrow Q) \land \{P \land e\} s; \text{ while } e \{ s \} \{Q\}\)

\((P \land \neg e \rightarrow Q) \land \{P \land e\} s \{P'\} \land \{P'\} \text{ while } e \{ s \} \{Q\}\)
Hoare Triples

\[(P \land \neg e \rightarrow Q) \land
\{P \land e\} \text{ s } \{P'\} \land
\{P'\} \text{ while } e \{ s \} \{Q\}
\]

\[
(P \land \neg e \rightarrow Q) \land
\{P \land e\} \text{ s } \{P'\} \land
(P' \land \neg e \rightarrow Q) \land
\{P' \land e\} \text{ s } \{P''\} \land
(P'' \land \neg e \rightarrow Q) \land
\ldots
\]

Invent infinitely-many conditions \(P^{(n)}\)
Loop Invariants

\[(P \land \neg e \rightarrow Q) \land \{P \land e\} \Rightarrow \{P'\} \land (P' \land \neg e \rightarrow Q) \land \{P' \land e\} \Rightarrow \{P''\} \land (P'' \land \neg e \rightarrow Q) \land \ldots\]

Simplest case: \(P^{(n)}\) is some **fixed condition** \(I\)

\[P \Rightarrow II \land (I \land \neg e \rightarrow Q) \land \{I \land e\} \Rightarrow \{I\}\]
Loop Invariants

New syntax for writing loop invariants:

\[
\{P\} \text{ while } \{I\} \; e \; \{s\} \; \{Q\}
\]

\[
P \rightarrow I \land \\
(I \land \neg e \rightarrow Q) \land \\
\{I \land e\} \; s \; \{I\}
\]

Weakest precondition computed from invariant:

\[
WP(Q) = I \land (I \land \neg e \rightarrow Q) \land (I \land e \rightarrow WP[s](I))
\]
Demo

Count to \( n \)

Search an array

Binary search
Exercises

Write down **loop invariants** for the following loops:

```
{ n = m ∧ n ≥ 0 ∧ i = 0 } while (n > 0) { i++; n--; }
{ i = m }
```

```
{ l ≥ 0 ∧ n = 1 } k = 0 while k < l:
  k += 1
  n *= 2
{ n = 2^l }
```

```
{ l < r } while rand():
  t = l
  l = r
  r = t
{ l ≠ r }
```
Invariant properties

If $I_1$ and $I_2$ are invariants, so is $I_1 \land I_2$; the reverse if false:

\[
\begin{align*}
\{ & 0 < n \land 0 < m \} \\
n & += m \\
m & += n \\
\{ & 0 < n \land 0 < m \}
\end{align*}
\]

You may need an $I$ that is stronger than $P$

\[
\begin{align*}
\{ & n = m \land n \geq 0 \land i = 0 \} \\
\text{while} \ (n > 0) & \{ \ i++; \ n--; \ } \\
\{ & i = m \}
\end{align*}
\]
Course Updates

Milestone Presentations and Assignment 4
Assignment 4

Use Dafny to verify several sorting algorithms:

- Bubble sort
- Insertion sort
- Merge sort

Proofs due March 5; do not change the specification!

Dafny available online but much faster if installed locally.
Milestone I

Milestone presentations on **March 3**; time limit like proposal

- What did you **get done**?
- What did you **learn in the process**?
- Where did you **deviate from the plan**?
- What is your **plan for Milestone II**?

Switch who presents if proposal was solo
Termination

Go away, halting problem
Must We Terminate?

Behavior of a statement given by a **triple**:

\[
\{P\} \ s \ \{Q\}
\]

If \( P \) true of a state, and \( s \) executed, then \( Q \) true after

If \( s \) doesn’t terminate, do we treat \( Q \) as true or false?

Doesn’t matter: **loops ought to terminate**
Proving Termination

Which of the following loops terminate?

\[
\begin{align*}
k &= 0 \\
\textbf{while} \ k < l: \\
&\quad k += 1 \\
&\quad n *= 2
\end{align*}
\]
\[
\{ \ l \leq r \ \} \\
\textbf{while} \ r - l > 1: \\
\quad \text{if} \ f(): \\
\quad \quad \ r = (r + l) \ \text{//} \ 2 \\
\quad \text{else:} \\
\quad \quad \ l = (r + l) \ \text{//} \ 2
\]
\[
\{ \ m \geq 0 \ \} \\
\textbf{while} \ m > 1: \\
\quad \text{if} \ m \ % \ 2 == 0: \\
\quad \quad \ m = m / 2 \\
\quad \text{else:} \\
\quad \quad \ m = 2*m + 1
\]
Termination

What does it mean that a loop terminates?
Loop must have a finite number of iterations

Idea: compute number of iterations for the loop
Problem: usually hard, requires inductive reasoning

Better idea: bound number of iterations for the loop
Usually easy by hand, often possible automatically
Bounding Iterations

Bound must **decrease** every iteration

Function that computes bound called the “measure” $M$

```
while e:
    \{ I \land e \}  while e:
    \{ M() = n \}

s   s
\{ I \} \{ M() < n \}
```

Can assume $I$ and $e$ to prove measure decreases
Higher Ordinals

Sometimes convenient to have non-integer measures

Lexicographic order, tree depth very common

\[
\text{while } m > 0: \\
\quad \text{if } n == 0: \\
\quad \quad m-- \\
\quad \quad n = f(m) \\
\quad \text{else:} \\
\quad \quad n--
\]

\((m, n)\) always decreases

Bound depends on “f”

Key property: cannot decrease infinitely many times
Measures Example

```python
k = 0
while k < l:
    k += 1
    n *= 2

{ l ≤ r }
while r - l > 1:
    if f():
        r = (r + l) // 2
    else:
        l = (r + l) // 2
```

```
{ n ≥ 0 }
while m > 0:
    if n == 0:
        m -= 1
        n = m
    else:
        n -= 1

(m, n) decreases
```

k decreases
r - l decreases
Exercises

Find loop measures for the following loops:

```python
{ m = 0 }
while n > 0:
    n = n / 2
    m++
```

```python
{ stack : List<T> }
while stack:
    print(f(stack.pop()))
```
Exercises

Find **loop measures** for the following loops:

```python
{ binary_search_tree(node) }
while True:
    if node.value == x:
        return node
    elif node.value < x:
        node = node.right
    else:
        node = node.left
```
Loop Verification

Verifying a loop requires **two inputs:**

Invariant $I$  
Measure $M$

With two constraints:

$$\{I \land e\} \implies \{I\}$$
$$\{I \land e \land M = m\} \implies \{M < m\}$$

Automatically discovering $I$ and $M$ is **impossible**.

It would solve the halting problem if it were possible!

**Though heuristics work quite well**
Next class:

Procedures

To do:

- Course feedback
- Milestone I presentation
- Assignment 4
Loops

What are the **weakest preconditions of loops**?

Imagining loops as infinitely-long sequences

**Loop invariants** for verification

What does not change over a loop iteration

**Proving termination** using decreasing measures

It can’t go below zero!
PROCEDURES

REUSING

P AND Q
Recursion, Again

Measures,

Again
SEPARATION

FRAME

PROBLEM
Next class:

Procedures

To do:

- Course feedback
- Milestone I presentation
- Assignment 4