1 Hoare Logic

The goal of Hoare logic is to provide a formal system for reasoning about program correctness. Hoare logic is based on the idea of a specification as a contract between the implementation of a function and its clients. The specification is made up of a precondition and a postcondition. The precondition is a predicate describing the condition the function relies on for correct operation; the client must fulfill this condition. The postcondition is a predicate describing the condition the function establishes after correctly running; the client can rely on this condition being true after the call to the function.

The implementation of a function is partially correct with respect to its specification if, assuming the precondition is true just before the function executes, then if the function terminates, the postcondition is true. The implementation is totally correct if, again assuming the precondition is true before function executes, the function is guaranteed to terminate and when it does, the postcondition is true. Thus total correctness is partial correctness plus termination.

Note that if a client calls a function without fulfilling its precondition, the function can behave in any way at all and still be correct. Therefore, if it is intended that a function be robust to errors, the precondition should include the possibility of erroneous input and the postcondition should describe what should happen in case of that input (e.g. a specific exception being thrown).

Hoare logic uses Hoare Triples to reason about program correctness. A Hoare Triple is of the form \{P\} S \{Q\}, where P is the precondition, Q is the
postcondition, and $S$ is the statement(s) that implement the function. The (total correctness) meaning of a triple $\{P\} \ S \ {Q}$ is that if we start in a state where $P$ is true and execute $S$, then $S$ will terminate in a state where $Q$ is true.

Consider the Hoare triple $\{x = 5\} \ x := x \times 2 \ {x > 0}$. This triple is clearly correct, because if $x = 5$ and we multiply $x$ by 2, we get $x = 10$ which clearly implies that $x > 0$. However, although correct, this Hoare triple is not a precise as we might like. Specifically, we could write a stronger postcondition, i.e. one that implies $x > 0$. For example, $x > 5 \&\& x < 20$ is stronger because it is more informative; it pins down the value of $x$ more precisely than $x > 0$. The strongest postcondition possible is $x = 10$; this is the most useful postcondition. Formally, if $\{P\} \ S \ {Q}$ and for all $Q$ such that $\{P\} \ S \ {Q}$, $Q \Rightarrow Q$, then $Q$ is the strongest postcondition of $S$ with respect to $P$.

We can reason in the opposite direction as well. If $\{P\} \ S \ {Q}$ and for all $P$ such that $\{P\} \ S \ {Q}$, $P \Rightarrow P$, then $P$ is the weakest precondition $wp(S, Q)$ of $S$ with respect to $Q$.

We can define a function yielding the weakest precondition with respect to some postcondition for assignments, statement sequences, and if statements, as follows:

$$wp(x := E, P) = [E/x]P$$
$$wp(S; T, Q) = wp(S, wp(T, Q))$$
$$wp(\text{if } B \text{ then } S \text{ else } T, Q) = B \Rightarrow wp(S, Q) \&\& \neg B \Rightarrow wp(T, Q)$$

A function can also be defined for the strongest postcondition with respect to some precondition, but this requires existential variables (to represent the old value of the variable being assigned) and is beyond the scope of these notes.

In order to verify the partial correctness of loops of the form while $b$ do $S$, we come up with an invariant $I$ such that the following conditions hold:

- $P \Rightarrow I$ : The invariant is initially true.
- $\{Inv \&\& B\} \ S \ {Inv}$ : Each execution of the loop preserves the invariant.
- $\{Inv \&\& \neg B\} \Rightarrow Q$ : The invariant and the loop exit condition imply the postcondition.
We can verify full correctness by coming up with an integer-valued variant function $v$ that fulfills the following conditions:

- $\text{Inv} \land \text{B} \Rightarrow v > 0$: If we are entering the loop body (i.e. the loop condition $B$ evaluates to true) and the invariant holds, then $v$ must be strictly positive.

- $\{\text{Inv} \land \text{B} \land v = V\} S \{v < V\}$: The value of the variant function decreases each time the loop body executes (here $V$ is a constant).

## 2 Proofs with Hoare Logic

Consider the WHILE program used in the previous lecture notes:

```
r := 1;
i := 0;
while $i < m$ do
   $r := r \times n$;
i := $i + 1$
```

We wish to prove that this function computes the $n$th power of $m$ and leaves the result in $r$. We can state this with the postcondition $r = n^m$.

Next, we need to determine the proper precondition. We cannot simply compute it with $wp$ because we do not yet know what the right loop invariant is—and in fact, different loop invariants could lead to different preconditions. However, a bit of reasoning will help. We must have $m \geq 0$ because we have no provision for dividing by $n$, and we avoid the problematic computation of $0^0$ by assuming $n > 0$. Thus our precondition will be $m \geq 0 \land kn > 0$.

We need to choose a loop invariant. A good heuristic for choosing a loop invariant is often to modify the postcondition of the loop to make it depend on the loop index instead of some other variable. Since the loop index runs from $i$ to $m$, we can guess that we should replace $m$ with $i$ in the postcondition $r = n^m$. This gives us a first guess that the loop invariant should include $r = n^i$.

This loop invariant is not strong enough (doesn’t have enough information), however, because the loop invariant conjoined with the loop exit condition should imply the postcondition. The loop exit condition is $i \geq m$, but we need to know that $i = m$. We can get this if we add $i \leq m$ to the loop invariant. In addition, for proving the loop body correct, it is convenient
to add $0 \leq i$ and $n > 0$ to the loop invariant as well. Thus our full loop invariant will be $r = n^i$ \&\& $0 \leq i \leq m$ \&\& $n > 0$.

In order to prove full correctness, we need to state a variant function for the loop that can be used to show that the loop will terminate. In this case $m - i$ is a natural choice, because it is positive at each entry to the loop and decreases with each loop iteration.

Our next task is to use weakest preconditions to generate proof obligations that will verify the correctness of the specification. We will first ensure that the invariant is initially true when the loop is reached, by propagating that invariant past the first two statements in the program:

```
{m \geq 0 \&\& n > 0}
    r := 1;
    i := 0;
{r = n^i \&\& 0 \leq i \leq m \&\& n > 0}
```

We propagate the loop invariant past $i := 0$ to get $r = n^0 \&\& 0 \leq 0 \leq m$ \&\& $n > 0$. We propagate this past $r := 1$ to get $1 = n^0 \&\& 0 \leq 0 \leq m$ \&\& $n > 0$. Thus our proof obligation is to show that:

$$m \geq 0 \&\& n > 0 \Rightarrow 1 = n^0 \&\& 0 \leq 0 \leq m \&\& n > 0$$

We prove this with the following logic:

$m \geq 0 \&\& n > 0$ by assumption
$1 = n^0$ because $n^0 = 1$ for all $n > 0$ and we know $n > 0$
$0 \leq 0$ by definition of $\leq$
$0 \leq m$ because $m \geq 0$ by assumption
$n > 0$ by the assumption above
$1 = n^0 \&\& 0 \leq 0 \leq m \&\& n > 0$ by conjunction of the above

We now apply weakest preconditions to the body of the loop. We will first prove the invariant is maintained, then prove the variant function decreases. To show the invariant is preserved, we have:

```
{r = n^i \&\& 0 \leq i \leq m \&\& n > 0 \&\& i < m}
    r := r \ast n;
    i := i + 1;
{r = n^i \&\& 0 \leq i \leq m \&\& n > 0}
```
We propagate the invariant past \( i := i + 1 \) to get \( r = n^{i+1} \land 0 \leq i + 1 \leq m \land n > 0 \). We propagate this past \( r := r \ast n \) to get: \( r \ast n = n^{i+1} \land 0 \leq i + 1 \leq m \land n > 0 \). Our proof obligation is therefore:

\[
\begin{align*}
r &= n^i \land 0 \leq i \leq m \land n > 0 \land i < m \land r \ast n = n^{i+1} \land 0 \leq i + 1 \leq m \land n > 0
\end{align*}
\]

We can prove this as follows:

\[
\begin{align*}
r &= n^i \land 0 \leq i \leq m \land n > 0 \land i < m & \text{by assumption} \\
r \ast n &= n^i \ast n & \text{by multiplying by } n \\
r \ast n &= n^{i+1} & \text{by definition of exponentiation} \\
0 \leq i + 1 & \text{because } 0 \leq i \\
i + 1 < m + 1 & \text{by adding 1 to inequality} \\
i + 1 \leq m & \text{by definition of } \leq \\
n > 0 & \text{by assumption} \\
r \ast n &= n^{i+1} \land 0 \leq i + 1 \leq m \land n > 0 & \text{by conjunction of the above}
\end{align*}
\]

We have a proof obligation to show that the variant function is positive when we enter the loop. The obligation is to show that the loop invariant and the entry condition imply this:

\[
\begin{align*}
r &= n^i \land 0 \leq i \leq m \land n > 0 \land i < m \\
\Rightarrow m - i & > 0
\end{align*}
\]

The proof is trivial:

\[
\begin{align*}
r &= n^i \land 0 \leq i \leq m \land n > 0 \land i < m & \text{by assumption} \\
i < m & \text{by assumption} \\
m - i & > 0 & \text{subtracting } i \text{ from both sides}
\end{align*}
\]

We also need to show that the variant function decreases. We generate the proof obligation using weakest preconditions:

\[
\begin{align*}
\{ r = n^i \land 0 \leq i \leq m \land n > 0 \land i < m \land m - i = V \} \\
r := r \ast n; \\
i := i + 1; \\
\{ m - i < V \}
\end{align*}
\]

We propagate the condition past \( i := i + 1 \) to get \( m - (i + 1) < V \). Propagating past the next statement has no effect. Our proof obligation is therefore:
\[ r = n^i \land 0 \leq i \leq m \land n > 0 \land i < m \land m - i = V \]
\[ \Rightarrow m - (i + 1) < V \]

Again, the proof is easy:

\begin{align*}
  r &= n^i \land 0 \leq i \leq m \land n > 0 \land i < m \land m - i = V & \text{by assumption} \\
  m - i &= V & \text{by assumption} \\
  m - i - 1 &= V & \text{by definition of } < \\
  m - (i + 1) &= V & \text{by arithmetic rules} \\
\end{align*}

Last, we need to prove that the postcondition holds when we exit the loop. We have already hinted at why this will be so when we chose the loop invariant. However, we can state the proof obligation formally:

\[ r = n^i \land 0 \leq i \leq m \land n > 0 \land i \geq m \]
\[ \Rightarrow r = n^m \]

We can prove it as follows:

\begin{align*}
  r &= n^i \land 0 \leq i \leq m \land n > 0 \land i \geq m & \text{by assumption} \\
  i &= m & \text{because } i \leq m \text{ and } i \geq m \\
  r &= n^m & \text{substituting } m \text{ for } i \text{ in assumption} \\
\end{align*}