Contents

List of Algorithms

1 Invariants and Variants

Invariants are properties that are true at the start and end of an iteration of a loop, with most programs with invariants having a pattern similar to:

```
// pre: preconditions
init
// I,V
while (test) {
  // I and test
  Body
  // I
}
// I and not test
// post
```

In order to be satisfied that a program works, we must check that `init` establishes I, `Body` maintains I, the loop terminates and I and not `test` implies the post-condition. Additionally, the variant V must be decreasing at each stage.

This can be expressed more formally using a series of Hoare triples (where $\{P\}\texttt{Prog}\{Q\}$ means that if `Prog` is executed from a state satisfying P, it will terminate in a state satisfying Q). If the following hold

- $\{pre\}\texttt{Init}\{I\}$
- $\{I \wedge test\}\texttt{Body}\{I\}$
- $I \wedge \neg test \Rightarrow post$
- $I \Rightarrow v \in \mathbb{N}$
- $\{I \wedge test \wedge v = V\}\texttt{Body}\{v < V\}$

then

$$\{pre\}\texttt{Init}; while(test)\texttt{Body}\{post\}$$

Most invariants can be found in the form $I \mathrel{\hat{=}} z = f(n) \wedge 0 \leq n \leq N$, where the loop condition is often $n < N$. When the loop terminates, $\neg(n < N) \Rightarrow n \geq N$, as the invariant still holds, $n \geq N \wedge n \leq N \Rightarrow n = N \Rightarrow f(n) = f(N)$. In this case, the variant is often $N - n$, and the program terminates when $N - n = 0$.

In some cases, the invariant makes use of an accumulating parameter, and is of the form $I \mathrel{\hat{=}} z \oplus f(n) \wedge 0 \leq n \leq N$ where n is decreasing. For example, to calculate X^N, it is often easier to write the result as $X^N = z \times X^n$, and when $n = 0$, z is the solution.

It is possible to write invariants for `for`-loops, however, the variables need to be substituted in.

```
Init // I[a/i]
for(i <- a until b) { // I ∧ a ≤ i < b
  Body // I[i + 1/i]
} // I[b/i]
```