Contract-based Software Development

Rasmus Guldborg Pedersen

January 2015

Overview

Invariants for loops and proof by induction

What is an invariant for a loop, and how can it be used to give a formal proof of a loop? How can we argue that a loop will terminate? Explain proof by induction and relate it to how to prove that a program assertion is a loop invariant.

Induction steps

Prove that $P(n)$ holds for all values of n. Where n is a natural number.

Base case

Prove for some value of n ($n = 0$ or $n = 1$).

Inductive step

Prove for $n + 1$.

Example

$$P(n) : 0 + 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

Example: Basis

$$P(0) : 0 = \frac{0(0+1)}{2}$$

Example: Inductive step

Assume $P(n)$ holds. Show $P(n+1)$ holds.

$$\begin{aligned}
(0+1+2+\cdots+n)+(n+1) &= \frac{(n+1)((n+1)+1)}{2} && (1)\\
&= \frac{(n+1)(n+2)}{2} && (2)\\
&= \frac{n(n+1)+2(n+1)}{2} && (3)\\
&= \frac{n(n+1)}{2}+(n+1) && (4)
\end{aligned}$$

Invariant (general)

A predicate describing some property that can be relied upon always to be true.

Loop Invariant

```
// { Q }
// S_0
// { P }
while(B) {
    // { P ∧ B }
    // S
    // { P }
}
// {P ∧ ¬ B ⇒ R }
```

Loop Invariant: Example

Algorithm for summing integers in a array.
$a[0] + a[1] + \ldots a[N-1] = (\Sigma i | 0 \leq i < N : a[i])$

Loop Invariant: Example

```
// { 0 ≤ N }
int n = 0;
int s = 0;
// { s = (Σ i | 0 ≤ i < n : a[i]) }
while (n != N) {
    // { s = (Σ i | 0 ≤ i < n : a[i]) ∧ n ≠ N }
    s = s + a[n];
    n = n + 1;
    // { s = (Σ i | 0 ≤ i < n : a[i]) }
}
// { s = (Σ i | 0 ≤ i < N : a[i]) ∧ n = N }
```

Loop Invariant: Example proof

Basis: $n = 1$

$a[0] = (\Sigma i | 0 \leq i < 1 : a[i])$

Inductive step: $n + 1$

$a[0] + a[1] + \ldots + a[n-1] + a[n] = (\Sigma i | 0 \leq i < n + 1 : a[i])$

Loop Invariant: Example proof

```
while (n != N) {
    s = s + a[n];
    // { s = (Σ i | 0 ≤ i < n + 1 : a[i]) }
    n = n + 1;
}
```

Loop Invariant: Termination

Function T such that loop execution ends when $T = 0$.
$T = N - n$ for the example.

The End

"Testing shows the presence, not the absence of bugs."
— Edsger W. Dijkstra