Loop Invariants

Vesal Vojdani

Department of Computer Science
University of Tartu

Formal Methods (2014)

Warm-Up

- Consider a simple loop-free program:

```
int succ(int x) {
    a = x + 1;
    if (a - 1 == 0)
        y = 1;
    else
        y = a;
    return y;
}
```

- Show that $y = x + 1$ at the return statement.

While Loops

- Recall the proof rule

$$\frac{(\!|\, \phi \wedge e \,|\!)\ C\ (\!|\, \phi \,|\!)}{(\!|\, \phi \,|\!)\ \text{while } e \text{ do } C\ (\!|\, \phi \wedge \neg e \,|\!)}$$

- Given a ψ as post-condition…
- How can we apply this rule?
- What is the WP of a while loop?

Termination?

- Weakest Liberal Preconditions

$$wp\,[\![S]\!]\,\psi \equiv wp\,[\![S]\!]\,true \land wlp\,[\![S]\!]\,\psi$$

- We did not care about this distinction
  - Termination is an outdated concept. ;)
  - Only loops have different definitions.

WP for while loops

- WP $[\![\text{while } e \text{ do } C]\!]\ \psi$?
- Unrolling the loop:

$$\begin{aligned} F_0 &= \text{while } e \text{ do skip} \\ F_i &= \text{if } e \text{ then } C \text{ ; } F_{i-1} \text{ else skip} \end{aligned}$$

- WP for "exiting the loop after at most i iterations in a state satisfying ψ":

$$\begin{aligned} L_0 &\equiv \psi \wedge \neg e \\ L_i &\equiv (\neg e \to \phi) \wedge (e \to \text{WP } [\![C]\!]\ L_{i-1}) \end{aligned}$$

WLP for while loops

- WLP $[\![\text{while } e \text{ do } C]\!]\ \psi$?
- Unrolling the loop:

$$F_0 = \text{while } e \text{ do skip}$$
$$F_i = \text{if } e \text{ then } C \text{ ; } F_{i-1} \text{ else skip}$$

- WLP for "if we exit the loop after at most i iterations, the resulting state satisfies ψ":

$$L_0 \equiv \psi$$
$$L_i \equiv (\neg e \rightarrow \phi) \wedge (e \rightarrow \text{WLP}\ [\![C]\!]\ L_{i-1})$$

WLP for while loops

- WLP for "if we exit the loop after at most i iterations, the resulting state satisfies ψ":

$$
\begin{aligned}
L_0 &\equiv \psi \\
L_i &\equiv (\neg e \rightarrow \phi) \wedge (e \rightarrow \text{WLP}\, [\![C]\!]\, L_{i-1})
\end{aligned}
$$

- We then define

$$\text{WLP}\, [\![\text{while } e \text{ do } C]\!]\, \psi = \forall i \in \mathbb{N} : L_i$$

- Not very practical…

Precondition of a While Loop

To push ψ up through while e do C:

1. Guess a potential invariant ϕ.
2. Make sure $\phi \land \neg e \implies \psi$.
3. Compute $\phi' = \text{WLP}\ [\![C]\!]\ \phi$.
4. Check that $\phi \land e \implies \phi'$.
5. Then, ϕ is a pre-condition for ψ.

$$\frac{(\!|\ \phi \land e\ |\!)\ C\ (\!|\ \phi\ |\!)}{(\!|\ \phi\ |\!)\ \text{while } e \text{ do } C\ (\!|\ \phi \land \neg e\ |\!)}$$

Proof Tableaux for Loops

$$
\begin{array}{l}
\quad (\!| \phi |\!) \\
\text{while } e \text{ do } \{ \\
\qquad (\!| \phi \wedge e |\!) \\
\qquad (\!| \mathrm{WLP}\, [\![C]\!]\ \phi |\!) \\
\quad C \\
\qquad (\!| \phi |\!) \\
\} \\
\quad (\!| \phi \wedge \neg E |\!) \\
\quad (\!| \psi |\!)
\end{array}
$$

Exercise 1

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

Guessing the invariant

- Doing a trace:

iteration	x	y	z	B
0	6	1	0	*true*
1	6	1	1	*true*
2	6	2	2	*true*
3	6	6	3	*true*
4	6	24	4	*true*
5	6	120	5	*true*
6	6	720	6	*false*
i		$i!$	i	

- Formulate hypothesis: $y = z!$

Proof obligations

Want to establish $\psi \equiv y = x!$.

1. Our invariant $\phi \equiv y = z!$
2. Check that $\phi \land \neg(z \neq x) \implies \psi$.

Proof obligations

Want to establish $\psi \equiv y = x!$.

1. Our invariant $\phi \equiv y = z!$
2. Check that $\phi \land \neg(z \neq x) \implies \psi$.
3. Compute WLP of loop body:

Proof obligations

Want to establish $\psi \equiv y = x!$.

1. Our invariant $\phi \equiv y = z!$
2. Check that $\phi \wedge \neg(z \neq x) \implies \psi$.
3. Compute WLP of loop body:

$$\phi' \equiv y \cdot (z + 1) = (z + 1)!$$

4. Check if $\phi \wedge z \neq x \implies \phi'$.

Proof obligations

Want to establish $\psi \equiv y = x!$.

1. Our invariant $\phi \equiv y = z!$
2. Check that $\phi \land \neg(z \neq x) \implies \psi$.
3. Compute WLP of loop body:

$$\phi' \equiv y \cdot (z + 1) = (z + 1)!$$

4. Check if $\phi \land z \neq x \implies \phi'$.
5. Continue WLP computation with ϕ.

Exercise 2: Minimal-Sum Section

- Given an integer array $a[0], a[1], \ldots, a[n-1]$.
- A section of a is a continuous piece $a[i], a[i+1], \ldots, a[j]$ with $0 \leqslant i \leqslant j < n$.
- Section sum: $S_{i,j} = a[i] + \cdots + a[j]$.
- A minimal-sum section is a section $a[i], \ldots, a[j]$ s.t. for any other $a[i'], \ldots, a[j']$, we have $S_{i,j} \leqslant S_{i',j'}$.

What to do?

- Compute the sum of the minimal-sum sections in linear time.
- Prove that the code is correct!
- For example...
  - $[-1, 3, 15, -6, 4, -5]$ is -7 for $[-6, 4, -5]$.
  - $[-2, -1, 3, -3]$ is -3 for $[-2, -1]$ or $[-3]$.

The Program

```
int minsum(int a[]) {
    k = 1;
    t = a[0];
    s = a[0];
    while (k != n) {
        t = min(t + a[k], a[k]);
        s = min(s,t);
        k = k + 1;
    }
    return s;
}
```

Post-conditions

- The value s is smaller than the sum of any section.

$$\phi_1 = \forall i, j : 0 \leqslant i \leqslant j < n \rightarrow s \leqslant S_{i,j}$$

- There is a section whose sum is s

$$\phi_2 = \exists i, j : 0 \leqslant i \leqslant j < n \wedge s = S_{i,j}$$

Trying to prove ϕ_1

- Suitable Invariant:

$$\phi_1 = \forall i, j : 0 \leqslant i \leqslant j < n \rightarrow s \leqslant S_{i,j}$$
$$I_1(s, k) = \forall i, j : 0 \leqslant i \leqslant j < k \rightarrow s \leqslant S_{i,j}$$

Trying to prove ϕ_1

- Suitable Invariant:

$$\phi_1 = \forall i, j : 0 \leqslant i \leqslant j < n \rightarrow s \leqslant S_{i,j}$$
$$I_1(s, k) = \forall i, j : 0 \leqslant i \leqslant j < k \rightarrow s \leqslant S_{i,j}$$

- Additional Invariant

$$I_2(t, k) = \forall i : 0 \leqslant i < k \rightarrow t \leqslant S_{i,k-1}$$

The Key Lemma

- In the end, we have to prove that

$$I_1(s, k) \wedge I_2(t, k) \wedge k \neq n$$
$$\Longrightarrow$$
$$I_1(\min(s, (\min(t + a[k], a[k])), k + 1)$$
$$\wedge$$
$$I_2(\min(t + a[k], a[k]), k + 1)$$

- This will require human intervention: proof-assistants.