Loop Invariant Proofs

CS 234

April 7, 2025

0 Introduction

This document contains examples of good proofs for the loop invariant problems in class. These are not the only ways to write good proofs.

These proofs may contain footnotes explaining different thought processes that occurred in their construction, to help show you how to think about writing proofs. Commentary may also be provided at the end about alternative approaches.

1 Proofs

```
1       def isSumEven(lst):
2           total = 0
3           i = 0
4           while i < len(lst):
5               total += lst[i]
6               i += 1
7           return total % 2 == 0
```

Theorem 1. *For all lists of integers* `lst`*, calling* `isSumEven(lst)` *returns a Boolean indicating whether* $\sum_{j=0}^{\texttt{len(lst)}-1} \texttt{lst[j]}$ *is even*[1]

Proof. This property is proven using the following loop invariant:

$$P(n) := \texttt{total} = \sum_{j=0}^{\texttt{i}-1} \texttt{lst[j]}$$

for the values of `total`, `i` from right before checking the loop guard for the n^{th} time[2] (0-indexed).[3]

Initialization From the start of the function until the guard is first checked in line 4, all the code does is set `total` and `i` to 0. This circumstance suffices to show that $P(0)$ holds because of the following identity:

$$\begin{aligned} \texttt{total} &= 0 && [\textit{line } 2] \\ &= \sum_{j=0}^{-1} \texttt{lst[j]} && [\Sigma\ \textit{def}] \\ &= \sum_{j=0}^{\texttt{i}-1} \texttt{lst[j]} && [\textit{line } 3] \end{aligned}$$

Maintenance Suppose that $P(n)$ holds for some iteration n and the loop guard continues to be satisfied such that an n^{th} iteration will occur.[4] Then

[1] There is a small lie to this theorem: the theorem is missing the caveat *if the function returns at all*. What we are proving is *partial* correctness. To prove *total* correctness we would also need to supply an argument that the program terminates.

[2] The fact that this is the n^{th} iteration does not really a play a role in the proof, nor many of the proofs we will write. However, it *can* play a role in principle—here that role is just played by `i` since $n =$ `i` as an invariant. In addition, tracking n helps us to see how this proof is just another form of induction.

[3] It is worth noting that this predicate uses *every* variable involved in the loop, which here are `i` and `total`. If a variable is being used, then it probably plays a role in the correct functioning of the code, and therefore should also play a role in the proof.

[4] The n^{th} iteration occurs here, not the $(n+1)^{th}$.

$\texttt{total} = \sum_{j=0}^{\texttt{i}-1} \texttt{lst}[\texttt{j}]$ must hold by the definition of $P(n)$, and $\texttt{i} < \texttt{len}(\texttt{lst})$ by the loop guard.

The new iteration starts with line 5, where `total` is updated to a new value. Calling the old value $\texttt{total}'$, we can derive the following identity of the new value `total`:

$$\begin{aligned}
\texttt{total} &= \texttt{total}' + \texttt{lst}[\texttt{i}] && [\textit{line } 5] \\
&= (\sum_{j=0}^{\texttt{i}-1} \texttt{lst}[\texttt{j}]) + \texttt{lst}[\texttt{i}] && [\textit{total}' \textit{ identity}] \\
&= \sum_{j=0}^{\texttt{i}} \texttt{lst}[\texttt{j}] && [\Sigma \textit{ def}]
\end{aligned}$$

Then in line 6, `i` is updated to a new value. Calling the old value $\texttt{i}'$, the new value `i` is set to $\texttt{i}' + 1$ so that $\texttt{i}' = \texttt{i} - 1$. Substituting this identity for $\texttt{i}'$ gives us `total` in terms of the new value `i`: $\texttt{total} = \sum_{j=0}^{\texttt{i}-1} \texttt{lst}[\texttt{j}]$.

This is the final state of the program before the loop guard is checked for the $(n+1)^{th}$ time, so $P(n+1)$ is satisfied.

Termination Suppose that $P(n)$ holds for some iteration n and the loop guard is not satisfied so that no more iterations will occur. Then $\texttt{total} = \sum_{j=0}^{\texttt{i}-1} \texttt{lst}[\texttt{j}]$ must hold by the definition of $P(n)$, and $\texttt{i} \geq \texttt{len}(\texttt{lst})$ by the loop guard. Moreover, since `i` is incremented 1 at a time until this point, it must be that `i` equals `len(lst)` exactly. This identity can be substituted into the identity of `total` to find that $\texttt{total} = \sum_{j=0}^{\texttt{len(lst)}-1} \texttt{lst}[\texttt{j}]$.

After leaving the loop, the program executes line 7 and returns a Boolean indicating whether the value of `total` is even. Because $\texttt{total} = \sum_{j=0}^{\texttt{len(lst)}-1} \texttt{lst}[\texttt{j}]$, this satisfies the desired property.

□