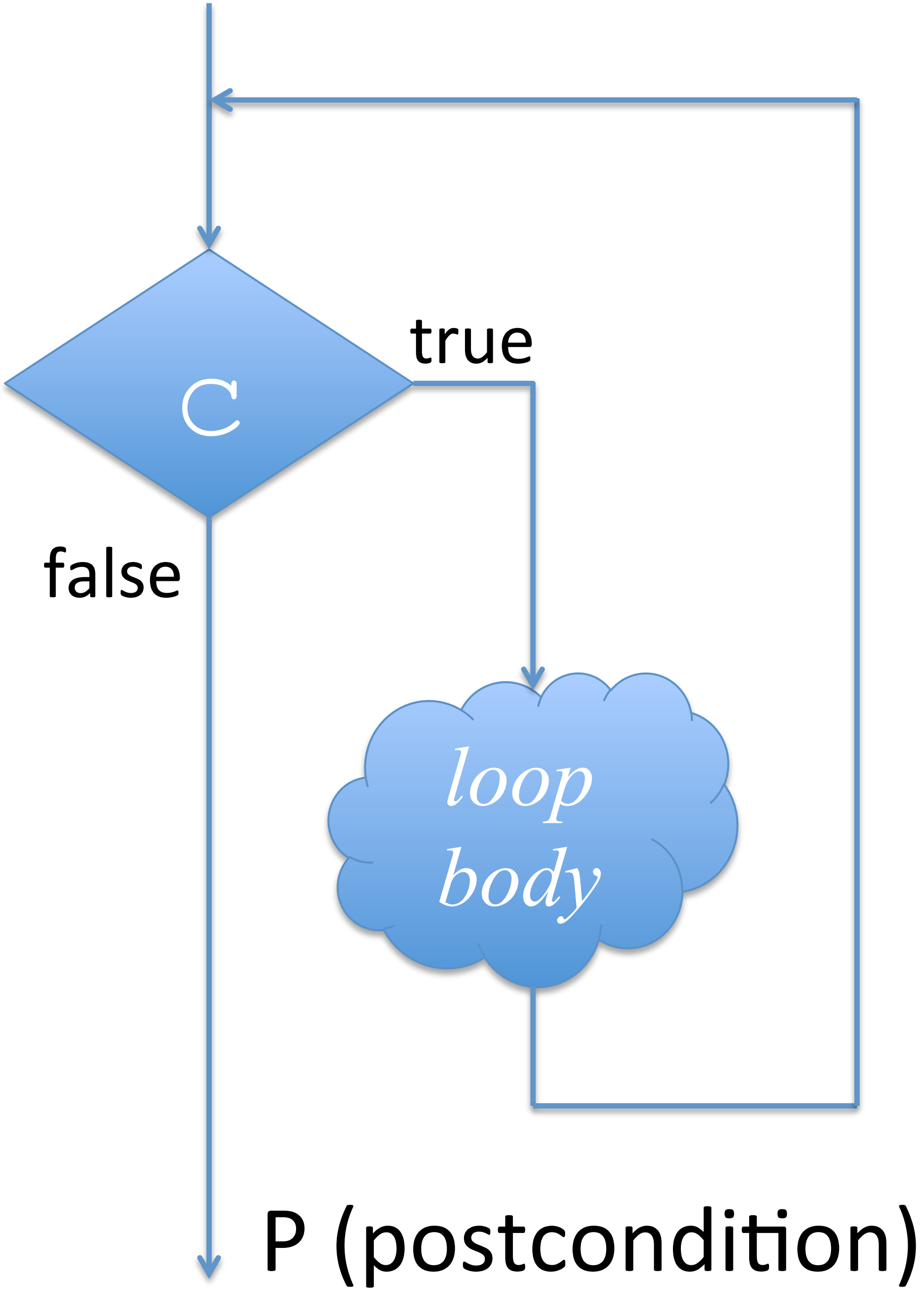
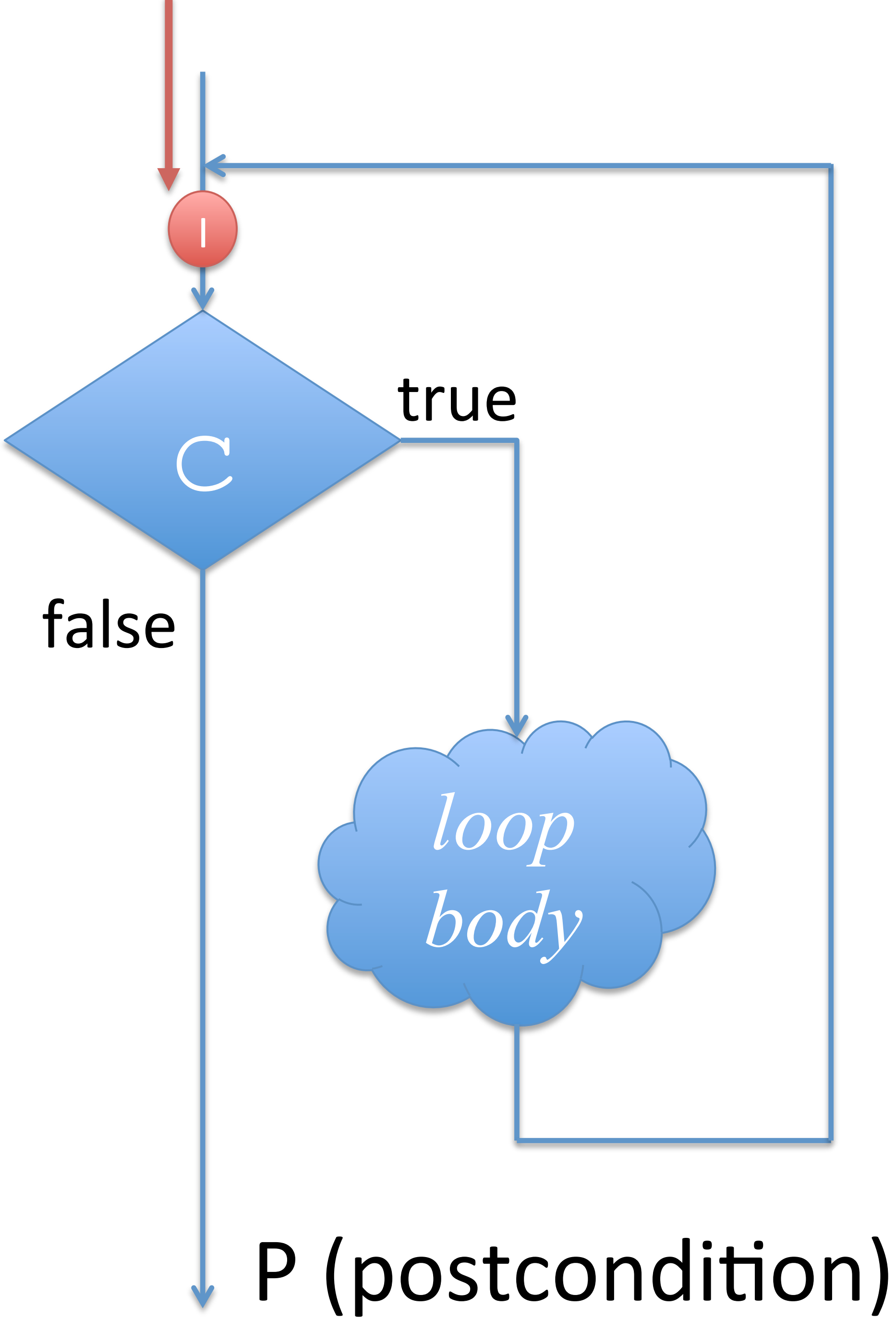
Loop Invariant

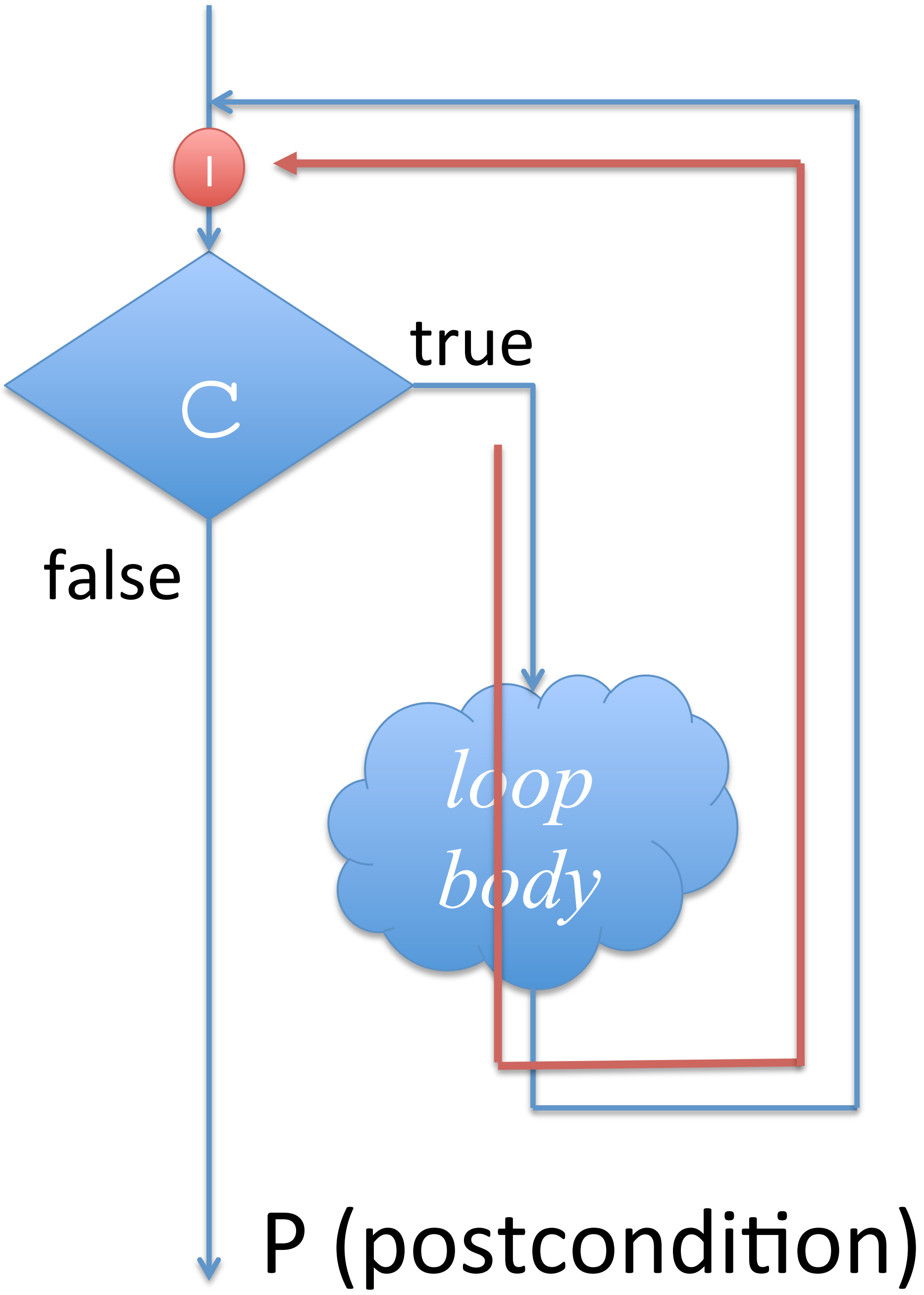
- Def’n: A boolean condition that is true immediately before every evaluation of the loop guard.
- It is true even if the loop runs 0 times (i.e. is skipped).
- It is true immediately before each evaluation of the loop guard, including the last evaluation if the loop terminates.
- It is true immediately after the loop terminates, if the loop terminates.



```
while (c)
//@loop_invariant I;
{
     loop body
}
//@assert P;
```

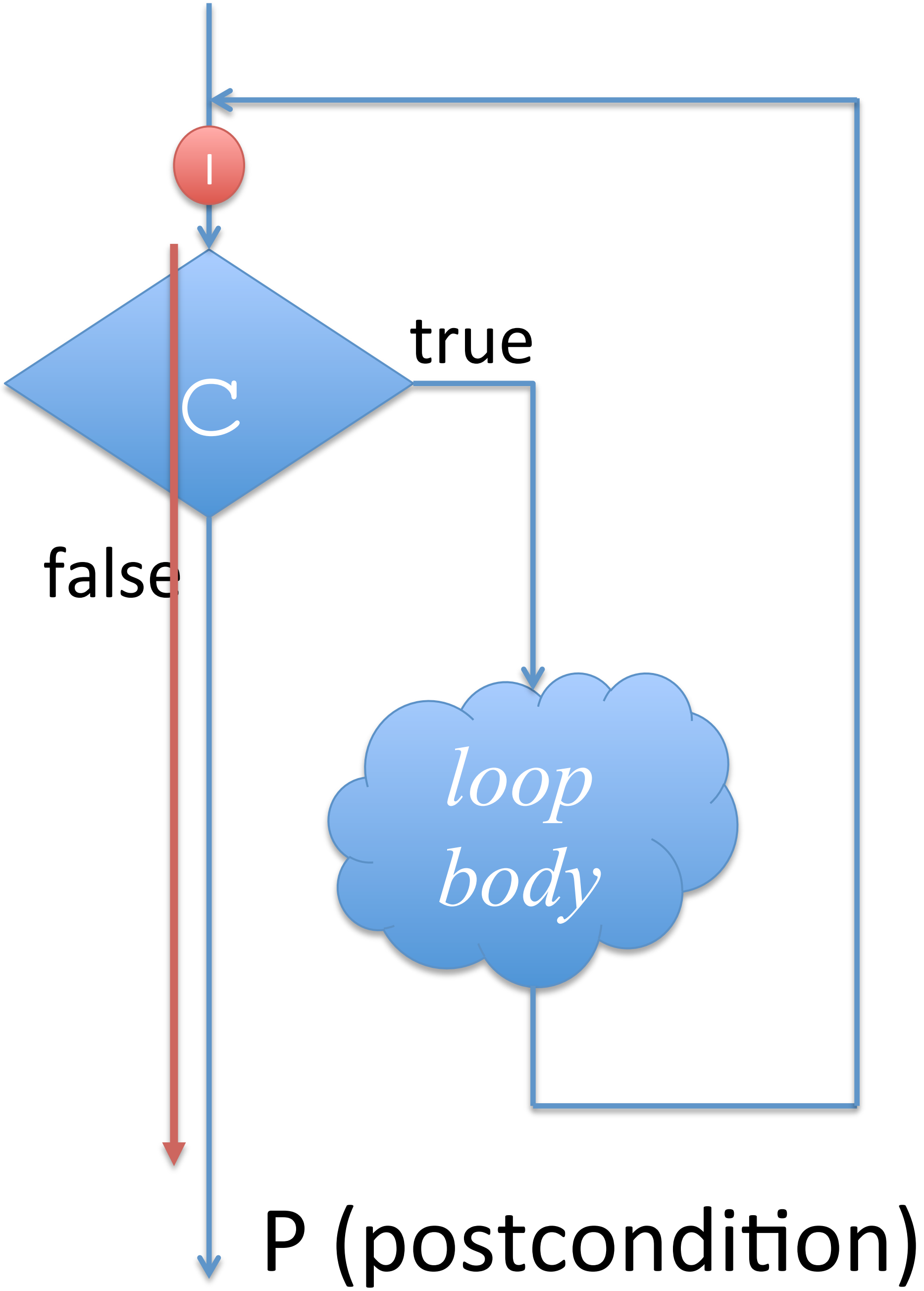


1. **INIT**
Show that the loop invariant I is true immediately before the first evaluation of the loop guard C.



2. **PRESERVATION**

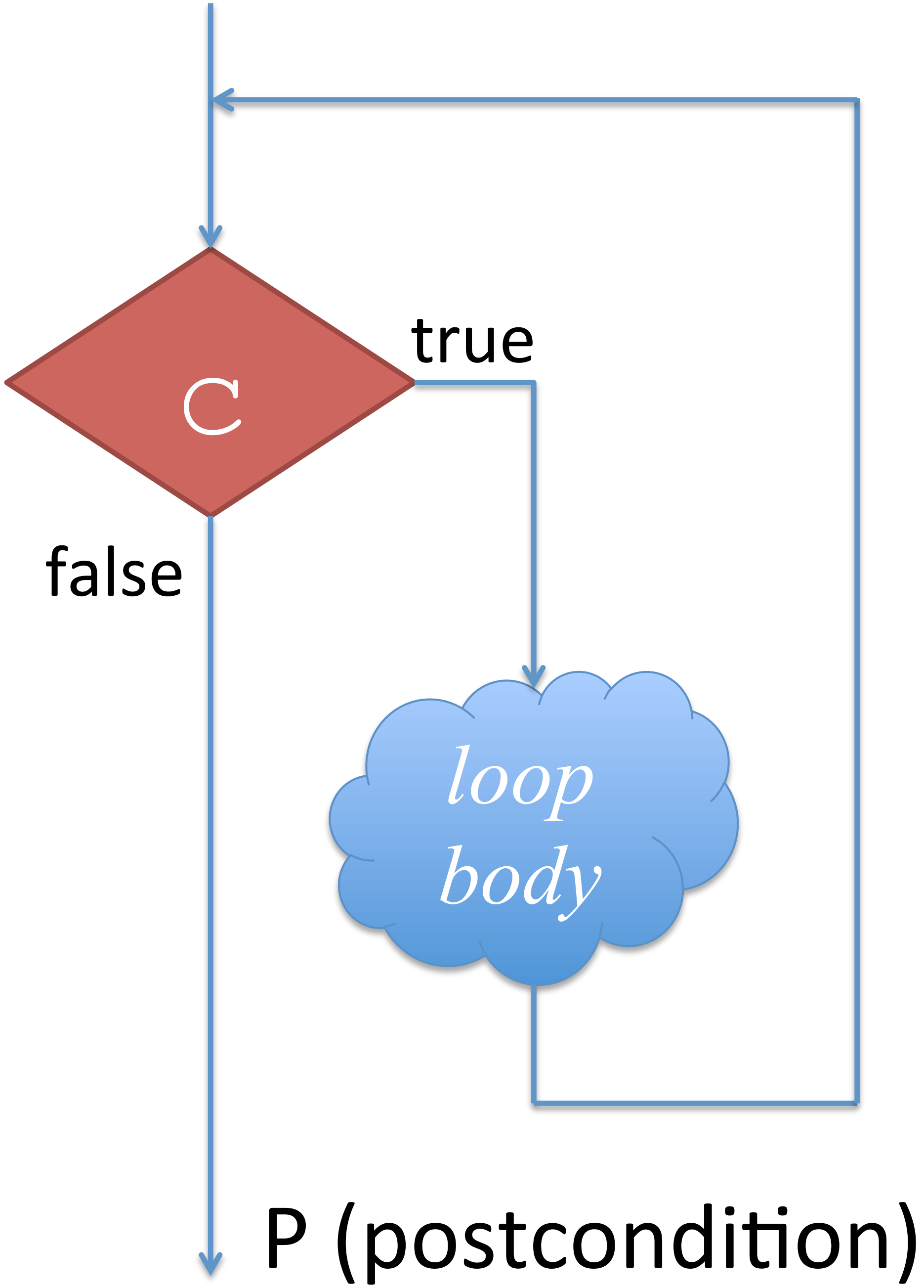
Show that if the loop invariant I is true immediately before the evaluation of the loop guard C, then I is true immediately before the next evaluation of the loop guard C.



3. **EXIT**

Once we have a valid loop invariant, we can show that the logical conjunction of the loop invariant I and the negation of the loop guard C implies the desired postcondition P:

I ^ ~ C → P



4. TERMINATION
Show that the loop will always terminate (i.e. that C must eventually be false).

Reasoning with a Loop Invariant

Given a loop with a loop guard C and a postcondition P, we can use the loop invariant I to reason that the postcondition must follow.

- We use step 1 to reason that loop invariant I is true immediately before first evaluation of C.

Reasoning with a Loop Invariant

- We use step 2 to reason that loop invariant I must be true at the end of the first iteration (since we’ve reasoned it is true at the start of the first iteration).

Reasoning with a Loop Invariant

- Since I was true at the end of the first iteration, it is also true at the start of the second iteration.
- We use step 2 to reason that loop invariant I must be true at the end of the second iteration (since we've reasoned it is true at the start of the second iteration).

Reasoning with a Loop Invariant

- Since I was true at the end of the second iteration, it is also true at the start of the third iteration.
- We use step 2 to reason that loop invariant I must be true at the end of the third iteration (since we've reasoned it is true at the start of the third iteration).

Reasoning with a Loop Invariant

… we can reason each iteration the same way until…

- Since I was true at the end of the next-to-last iteration, it is also true at the start of the last iteration.
- We use step 2 to reason that loop invariant I must be true at the end of the last iteration (since we've reasoned it is true at the start of the last iteration).

Reasoning with a Loop Invariant

- We use step 3 to reason about what happens when we exit the loop (step 4 ensures we will do so eventually).
- After the last iteration, C is now false, but I must be true (since I was true at the end of the last iteration).
- Once we know we have a proper loop invariants, we can use it to show that the conjuction of I ^ ~C implies P to argue that the desired postcondition holds.

Reasoning with a Loop Invariant

- Note that this reasoning works even if the loop executes 0 times. (step 2 is vacuous)
- Note that step 2 is used to reason about EVERY single iteration using the same logic. Step 2 acts as a generalization so we can reason about every execution of this loop, no matter how many times it will run.