Loop Invariant and While Loop Rule

- What should be the $wp(W, q)$ for $W \equiv \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}$? Here let's assume that W is error-free.
  - Denote w_i as the weakest precondition where loop W runs exactly i iterations.
    - If we never enter the loop body, then $wp(W, q) = w_0 = \neg B \wedge q$
    - If W runs exactly one iteration, then $wp(W, q) = w_1 = B \wedge wp(S, w_0)$
    - If W runs exactly two iterations, then $wp(W, q) = w_2 = B \wedge wp(S, w_1)$
    - …
    - If W runs exactly $k > 0$ iterations, then $wp(W, q) = w_k = B \wedge wp(S, w_{k-1})$
  - In general, we cannot predict how many iterations are needed for a loop, thus $wp(W, q) \equiv w_0 \vee w_1 \vee w_2 \ldots \vee w_k$ where k is the is the number of iterations W being executed, and k can be arbitrarily large. In other words, we cannot express $wp(W, q)$ in a finite expression.
  - The same problem also happens when we calculate $sp(p, W)$.

- Since we cannot calculate $wp(W, q)$ exactly, all we can do is to approximate it. In other words, so that we can show the correctness of a loop, we want to find a $p \Rightarrow wp(W, q)$, and p is not much stronger than $wp(W, q)$.
  - Here is an idea: if we can find a $p \Rightarrow w_i$ for each i, then $p \Rightarrow w_0 \vee w_1 \vee w_2 \ldots \vee w_k$.

- A **loop invariant** for $W \equiv \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}$ is a predicate p such that $\vDash \{p \wedge B\}\ S\ \{p\}$. It follows that $\vDash \{p\}\ W\ \{p \wedge \neg B\}$.
  - In other words, a loop invariant is a predicate that is True, before and after each iteration of the loop.
  - To indicate the loop invariant, we write an additional clause in the program: $W \equiv \{\textbf{inv}\ p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}$.

- **While Loop Rule:**
  1. $\{p \wedge B\}\ S\ \{p\}$
  2. $\{p\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \ \{p \wedge \neg B\}$ loop 1

  - The precondition p and postcondition $p \wedge \neg B$ are not the weakest precondition and the strongest postcondition, but they are the "best" precondition and postcondition we can find for a provable triple.

1. Which of the following predicate p can be a loop invariant for $W \equiv \{\textbf{inv}\ p\}\ \textbf{while}\ k < n\ \textbf{do}\ k := k + 1; s := s + k\ \textbf{od}$
   a. $p \equiv T$ Yes.
   b. $p \equiv F$ Yes.
   c. $p \equiv 0 \le k \le n$ Yes, $p \wedge B \Leftrightarrow 0 \le k < n$, so after $k \coloneqq k + 1$, we still have p

- From the above example, we can see that not all loop invariants provide us useful information: it cannot be too weak or too strong. If we have a loop $W \equiv \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}$, and we need $\vDash \{p_0\}\ W\ \{q\}$, then we need a loop invariant p such that:
  1) $p_0 \Rightarrow p$
  2) $\vDash \{p \wedge B\}\ S\ \{p\}$
  3) $p \wedge \neg B \Rightarrow q$

  If we consider the wlp and sp of the loop and loop body, then we need a loop invariant p such that:
  1) $p \Rightarrow wlp(W, p \wedge \neg B)$

2) $sp(p, W) \Rightarrow p \wedge \neg B$
3) $p \wedge B \Rightarrow wlp(S, p)$
4) $sp(p \wedge B, S) \Rightarrow p$

In future classes, we will discuss more on how to find a good loop invariant.

2. Show that p is a loop invariant for $W \equiv \{\mathbf{inv}\ p\}\ \mathbf{while}\ k < n\ \mathbf{do}\ k := k + 1; s := s + k\ \mathbf{od}$, where $p \equiv 0 \leq k \leq n \wedge s = sum(0, k)$, and $sum(0, k) \equiv \sum_{i=0}^{k} i$.

- We need to prove triple $\{p \wedge k < n\}\ S\ \{p\}$. We can create the following proof.

1. $\{p[s + k \ / \ s]\}\ \ s \coloneqq s + k\ \{p\}$ backward assignment
2. $\{p[s + k \ / \ s]\ [k + 1 \ / \ k]\}\ k \coloneqq k + 1\ \{p\ [s + k \ / \ s]\}$ backward assignment
3. $\{p[s + k \ / \ s]\ [k + 1 \ / \ k]\}\ k \coloneqq k + 1; s \coloneqq s + k\ \{p\}$ sequence 2,1
4. $p \wedge k < n \Rightarrow p[s + k \ / \ s]\ [k + 1 \ / \ k]$ predicate logic

$p \wedge k < n \Leftrightarrow 0 \leq k < n \wedge s = sum(0, k)$

$p[s + k \ / \ s]\ [k + 1 \ / \ k]\ \equiv \big(0 \leq k \leq n \wedge s + k = sum(0, k)\big)[k + 1 \ / \ k]$

$\equiv 0 \leq (k + 1) \leq n \wedge s + k + 1 = sum(0, k + 1)$

5. $\{p \wedge k < n\}\ k \coloneqq k + 1; s \coloneqq s + k\ \{p\}$ strengthen precondition 4,3
6. $\{\mathbf{inv}\ p\}\ W\ \{p \wedge k \geq n\}$ loop 5

3. Prove the following program under partial correctness.

$\{n \geq 0\}$
$k \coloneqq 0;\ \ s \coloneqq 0;$
$\{\mathbf{inv}\ p_1 \equiv 0 \leq k \leq n \wedge s = sum(0\ , k)\}$
$\mathbf{while}\ k < n\ \mathbf{do}$
$\quad s \coloneqq s + k + 1;\ \ k \coloneqq k + 1$
$\mathbf{od}$
$\{s = sum\ (0, n)\}$

- Informally, to prove the above program we need to prove the following triples/predicates:

$\{n \geq 0\}\ k \coloneqq 0;\ \ s \coloneqq 0\ \{p_1\}$
$\{p_1 \wedge\ k < n\}\ s \coloneqq s + k + 1;\ \ k \coloneqq k + 1\ \{p_1\}$
$p_1 \wedge k \geq n \Rightarrow s = sum(0, n)$

We can create the following proof.

1. $\{n \geq 0\}\ k \coloneqq 0\ \{n \geq 0 \wedge k = 0\}$ forward assignment
2. $\{n \geq 0 \wedge k = 0\}\ s \coloneqq 0\ \{n \geq 0 \wedge k = 0\ \wedge\ s = 0\}$ forward assignment
3. $\{n \geq 0\}\ k \coloneqq 0;\ \ s \coloneqq 0\ \{n \geq 0 \wedge k = 0 \wedge s = 0\}$ sequence 1,2
4. $n \geq 0 \wedge k = 0 \wedge s = 0 \rightarrow p_1$ predicate logic

Where $p_1 \equiv\ 0 \leq k \leq n\ \wedge\ s = sum\ (0\ , k\ )$

5. $\{n \geq 0\}\ k \coloneqq 0;\ \ s \coloneqq 0\ \{p_1\}$ weaken postcondition 3,4
6. $\{p_1[k + 1 \ / \ k]\}\ k \coloneqq k + 1\ \{p_1\}$ backward assignment
7. $\{p_1[k + 1 \ / \ k][s + k + 1 \ / \ s]\}\ s \coloneqq s + k + 1\ \{p_1[k + 1 \ / \ k]\}$ backward assignment
8. $\{p_1[k + 1 \ / \ k][s + k + 1 \ / \ s]\}\ S_1\ \{p_1\}$ sequence 7,6

Where $S_1 \equiv s \coloneqq s + k + 1;\ \ k \coloneqq k + 1$

9. $p_1 \wedge\ k < n \rightarrow p_1[k + 1 \ / \ k][s + k + 1 \ / \ s]$ predicate logic

$p_1 \wedge\ k < n \Leftrightarrow 0 \le k < n\ \wedge\ s = sum\,(0\,,k)$
$p_1[k+1\,/\,k][s+k+1\,/\,s] \equiv \big(0 \le k+1 \le n \wedge s = sum(0, k+1)\big)\,[s+k+1\,/\,s]$
$\equiv 0 \le k+1 \le n \wedge s+k+1 = sum(0, k+1)$

10. $\{p_1 \wedge\ k < n\}\, S_1\, \{p_1\}$ — strengthen precondtion 9, 8
11. $\{\mathbf{inv}\ p_1\}\ W\ \{p_1 \wedge k \ge n\}$ — loop 10
   # Where $W \equiv \mathbf{while}\ k < n\ \mathbf{do}\ S_1\ \mathbf{od}$

12. $\{n \ge 0\}\ k \coloneqq 0;\ s \coloneqq 0; W\ \{p_1 \wedge k \ge n\}$ — sequence 5,11
13. $p_1 \wedge\ k \ge n \rightarrow s = sum\,(0, n)$ — predicate logic
   # $p_1 \wedge\ k \ge n \Leftrightarrow 0 \le k = n\ \wedge\ s = sum\,(0\,,k)$

14. $\{n \ge 0\}\ k \coloneqq 0;\ s \coloneqq 0; W\ \{sum\,(0, n)\}$ — weaken postcondition 12, 13

Full Proof Outlines

- A formal proof can be very long and contains repetitive information. A ***proof outline*** is a way to write out all the information that you would need to generate a full formal proof, but with less repetition, so they're much shorter, and they don't mask the overall structure of the program the way a full proof does. Before studying how to write a ***full proof outline***, let us look at an example. Here we give the full proof outline for the program we proved in Question 3; I use a different color to show the parts that are added to the program.

4. Give a full proof outline for the program in Question 3.

$\{n \ge 0\}$
$k \coloneqq 0; \{n \ge 0 \wedge k = 0\}\ s \coloneqq 0;\ \{n \ge 0 \wedge k = 0 \wedge s = 0\}$
$\{\mathbf{inv}\ p_1 \equiv 0 \le k \le n \wedge s = sum(0\,,k)\}$
$\mathbf{while}\ k < n\ \mathbf{do}$
    $\{p_1 \wedge\ k < n\}$
    $\{p_1[k+1\,/\,k][s+k+1\,/\,s]\}\ s \coloneqq s+k+1; \{p_1[k+1\,/\,k]\}\ k \coloneqq k+1\ \{p_1\}$
$\mathbf{od}$
$\{p_1 \wedge k \ge n\}$
$\{s = sum\,(0, n)\}$

- To get a full proof outline, we annotate program statements with their preconditions and postconditions, so that every statement in the program is part of one or more correctness triples. ***Every triple must be provable using the proof rules***.

- Here are the rules to create proof outlines for individual statements:
  - Assignment and skip statements are handled just as they are in proof rules:
    - $\{p\}\ v \coloneqq e\ \{q\}$
    - $\{p\}\ \mathbf{skip}\ \{p\}$
  - Sequence statement that combines $\{p\}\ S_1\ \{q\}$ and $\{q\}\ S_2\ \{r\}$ and getting $\{p\}\ S_1; S_2\ \{r\}$ is written as:
    - $\{p\}\ S_1;\ \{q\}\ S_2\ \{r\}$
  - To express the while loop rule that loops through $\{p \wedge B\}\ S\ \{p\}$ to get $\{\mathbf{inv}\ p\}\ \mathbf{while}\ B\ \mathbf{do}\ S\ \mathbf{od}\ \{p \wedge \neg B\}$, we write:
    - $\{\mathbf{inv}\ p\}\ \mathbf{while}\ B\ \mathbf{do}\ \{p \wedge B\}\ S\ \{p\}\ \mathbf{od}\ \{p \wedge \neg B\}$

  - For conditional rule 1 we write:

- $\{p\}$ **if** B **then** $\{p \wedge B\}\ S_1\ \{q_1\}$ **else** $\{p \wedge \neg B\}\ S_2\ \{q_2\}$ **fi** $\{q_1 \vee q_2\}$

- For conditional rule 2 we write:
  - $\{(B \rightarrow p_1) \wedge (\neg B \rightarrow p_2)\}$ **if** B **then** $\{p_1\}\ S_1\ \{q_1\}$ **else** $\{p_2\}\ S_2\ \{q_2\}$ **fi** $\{q_1 \vee q_2\}$
- For nondeterministic conditional rule we write:
  - $\{p\}$ **if** $B_1 \rightarrow \{p \wedge B_1\}\ S_1\ \{q_1\} \ \square\ B_2 \rightarrow \{p \wedge B_2\}\ S_2\ \{q_2\}$ **fi** $\{q_1 \vee q_2\}$

- To strengthen the precondition of $\{p\}\ S\ \{q\}$ with $p_1 \Rightarrow p$, we write:
  - $\{p_1\}\{p\}\ S\ \{q\}$
- To weaken the postcondition of $\{p\}\ S\ \{q\}$ to $q \Rightarrow q_1$, we write:
  - $\{p\}\ S\ \{q\}\{q_1\}$