Interference freedom - example

1 Problem

$A = (a : \mathbb{Z}^n, b : \mathbb{Z}^n)$
$Pre = (a = a')$
$Post = (a = a' \wedge b = a')$

Given the following program with intemediate assertions of the total correctness proof outline. Show the interference freedom!

$i, j := 1, 1;$
$\{a = a' \wedge i = 1 \wedge j = 1\}$
parbegin $S_1 \| S_2$ **parend**

$\{Inv\}$
S_1 :
 	
$\{Inv\}$
while $i \leq n$ do
  $\{Inv \wedge i \leq n\}$
  await $i = j$ then
    $x, i := a[i], i + 1$
  ta
  $\{Inv\}$
do
$\{Inv \wedge i = n + 1\}$
	$\{Inv\}$
	S_2 :
	
	$\{Inv\}$
	while $j \leq n$ do
	  $\{Inv \wedge j \leq n\}$
	  await $i > j$ then
	    $b[j], j := x, j + 1$
	  ta
	  $\{Inv\}$
	do
	$\{Inv \wedge j = n + 1\}$

The invariant of the loops is given:
$Inv = (a = a' \wedge 0 \leq i - 1 \leq j \leq i \leq n + 1 \wedge \forall k \in [1..j - 1] : b[k] = a[k]) \wedge (i > j \Rightarrow x = a[i - 1]))$
$i : \mathbb{N}$ and $j : \mathbb{N}$ are auxiliary variables of the program.
The variant function of S_1 is $n + 1 - i$, whereas the variant function of S_2 is $n + 1 - j$.

1.1 Interference freedom proof - part 1

Interference freedom means that none of the statements in the parallel block invalidates the proof outline assertions of any other component. Now let us consider the await statment of S_1 and prove that it does not invalidate the desired behavior of S_2, meaning that execution of
await $i = j$ then $x, i := a[i], i + 1$ **ta** (its precondition is $Inv \wedge i \leq n$)
does not falsifies the assertions of S_2's proof outline. First write down the conditions that are needed to prove:

1. $\{\{(Inv \wedge i \leq n) \wedge Inv\}\}$ **await** $i = j$ then $x, i := a[i], i + 1$ **ta** $\{\{Inv\}\}$
2. $\{\{(Inv \wedge i \leq n) \wedge Inv\}\}$ **await** $i = j$ then $x, i := a[i], i + 1$ **ta** $\{\{Inv\}\}$
3. $\{\{(Inv \wedge i \leq n) \wedge (Inv \wedge j \leq n)\}\}$ **await** $i = j$ then $x, i := a[i], i + 1$ **ta** $\{\{Inv \wedge j \leq n\}\}$
4. $\{\{(Inv \wedge i \leq n) \wedge (Inv \wedge j = n+1)\}\}$ **await** $i = j$ then $x, i := a[i], i+1$ **ta** $\{\{Inv \wedge j = n+1\}\}$
5. We use the variant function $n + 1 - j$ to show that S_2 terminates. To show that the guarded assignment of S_1 does not interfere with any proof of S_2, we must show that the await statement of S_1 does not increase the value of the variant function of the loop of S_2:
$\{\{(Inv \wedge i \leq n) \wedge (n+1-j = t_0)\}\}$ **await** $i = j$ then $x, i := a[i], i+1$ **ta** $\{\{n+1-j \leq t_0\}\}$

Let us notice that it is sufficient to prove only the first one. Why? The await statement does not change the values of n and j. If $j \leq n$ holds before the execution, obviously it will be also true after the execution of the await statement. The same goes for $j = n+1$ and $n+1-j = t_0$.

Instead of proving the first condition, by the rule of the await statement it is sufficient to prove that
$\{\{(Inv \wedge i \leq n) \wedge i = j\}\}$ $x, i := a[i], i + 1$ $\{\{Inv\}\}$

Moreover, by the definition of the weakest precondition, it is sufficient to prove that
$(Inv \wedge i \leq n \wedge i = j) \Rightarrow wp(x, i := a[i], i + 1, Inv)$
since $wp(x, i := a[i], i + 1, Inv)$ describes the set of all initial states that will guarantee termination of $x, i := a[i], i + 1$ in a state satisfying Inv.
Let us calculate the condition $wp(x, i := a[i], i + 1, Inv)$ and prove that it can be deduced from $Inv \wedge i \leq n \wedge i = j$.

$wp(x, i := a[i], i + 1, Inv) = (a = a' \wedge 0 \leq i \leq j \leq i + 1 \leq n + 1 \wedge \forall k \in [1..j - 1]\colon b[k] = a[k]) \wedge (i + 1 > j \Rightarrow a[i] = a[i]) \wedge i \in [1..n])$

Notice that $i \in [1..n]$ has to be added because this condition guarantees that the assignment $x := a[i]$ does not abort.

1.1.1 Proof - Part 1

We want to prove:
$(Inv \wedge i \leq n \wedge i = j) \implies (a = a' \wedge 0 \leq i \leq j \leq i+1 \leq n+1 \wedge \forall k \in [1..j-1]\colon b[k] = a[k]) \wedge (i+1 > j \Rightarrow a[i] = a[i]) \wedge i \in [1..n])$

- $a = a'$
  This condition is true, because it is included in the invariant.
- $\forall k \in [1..j-1]\colon b[k] = a[k]$
  This condition is true, because it is included in the invariant.
- $i+1 > j \Rightarrow a[i] = a[i]$
  The logical implication is true, because the consequent $a[i] = ai[]$ is true. ($i+1 > j$ also holds, anyway, since the guard $i = j$ of the await holds.)
- $i \in [1..n]$ that is equivalent to $1 \leq i \wedge i \leq n$
  $1 \leq i$ holds since $0 \leq i-1$ (by Inv) and $i \leq n$ holds since it was included in the precondition of the await.
- $0 \leq i \leq j \leq i+1 \leq n+1$
  This is the only remaining condition we have to prove.
  - $0 \leq i$
    We already proved that $1 \leq i$. Since $1 \leq i$ holds, the condition $0 \leq i$ is obviously true.
  - $i \leq j$
    This holds, since $i = j$.
  - $j \leq i+1$
    j cannot be greater than $i+1$. This holds, since $j = i$.
  - $i+1 \leq n+1$
    This holds, since $i \leq n$ is true.

1.2 Interference freedom proof - part 2

Now let us consider the await statment of S_2 and prove that it does not invalidate the desired behavior of S_1, meaning that execution of
await $i > j$ **then** $b[j], j := x, j + 1$ **ta** (its precondition is $Inv \wedge j \leq n$)
does not falsifies the assertions of S_1's proof outline. First write down the conditions that are needed to prove:

1. $\{\{(Inv \wedge j \leq n) \wedge Inv\}\}$ **await** $i > j$ **then** $b[j], j := x, j + 1$ **ta** $\{\{Inv\}\}$
2. $\{\{(Inv \wedge j \leq n) \wedge Inv\}\}$ **await** $i > j$ **then** $b[j], j := x, j + 1$ **ta** $\{\{Inv\}\}$
3. $\{\{(Inv \wedge j \leq n) \wedge (Inv \wedge i \leq n)\}\}$ **await** $i > j$ **then** $b[j], j := x, j+1$ **ta ta** $\{\{Inv \wedge i \leq n\}\}$
4. $\{\{(Inv \wedge j \leq n) \wedge (Inv \wedge i = n+1)\}\}$ **await** $i > j$ **then** $b[j], j := x, j+1$ **ta** $\{\{Inv \wedge i = n+1\}\}$
5. We use the variant function $n + 1 - i$ to show that S_1 terminates. To show that the guarded assignment of S_2 does not interfere with any proof of S_1, we must show that the await statement of S_2 does not increase the value of the variant function of the loop of S_1:
$\{\{(Inv \wedge j \leq n) \wedge (n+1-i = t_0)\}\}$ **await** $i > j$ **then** $b[j], j := x, j+1$ **ta** $\{\{n+1-i \leq t_0\}\}$

Let us notice that it is sufficient to prove only the first one. Why? The await statement does not change the values of n and i. If $i \leq n$ holds before the execution, obviously it will be also true after the execution of the await statement. The same go for $i = n + 1$ and $n + 1 - i = t_0$.

Instead of proving the first condition, by the rule of the await statement it is sufficient to prove that
$\{\{(Inv \wedge j \leq n) \wedge i > j\}\}$ $b[j], j := x, j + 1$ $\{\{Inv\}\}$

Moreover, by the definition of the weakest precondition, it is sufficient to prove that
$(Inv \wedge j \leq n \wedge i > j) \Rightarrow wp(b[j], j := x, j + 1, Inv)$
since $wp(b[j], j := x, j + 1, Inv)$ describes the set of all initial states that will guarantee termination of $b[j], j := x, j + 1$ in a state satisfying Inv.
Let us calculate the condition $wp(b[j], j := x, j + 1, Inv)$ and prove that it can be deduced from $Inv \wedge j \leq n \wedge i > j$.

$wp(b[j], j := x, j + 1, Inv) = (a = a' \wedge 0 \leq i - 1 \leq j + 1 \leq i \leq n + 1 \wedge \forall k \in [1..j]\colon b[k] = a[k]) \wedge (i > j + 1 \Rightarrow x = a[i - 1]) \wedge j \in [1..n])$

Notice that $j \in [1..n]$ has to be added because this condition guarantees that the assignment $b[j] := x$ does not abort.

1.2.1 Proof - Part 2

We want to prove:
$(Inv \wedge j \leq n \wedge i > j) \Longrightarrow (a = a' \wedge 0 \leq i-1 \leq j+1 \leq i \leq n+1 \wedge \forall k \in [1..j]\colon b[k] = a[k]) \wedge (i >$

$j+1 \Rightarrow x = a[i-1]) \wedge j \in [1..n])$

Notice that $i-1 \leq j$ and $j < i$ together imply that $i-1 = j$.

- $a = a'$
  This condition is true, because it is included in the invariant.
- $j \in [1..n]$ that is equivalent to $1 \leq j \wedge j \leq n$
  $1 \leq j$ holds since j is a natural number and $j \leq n$ holds since it was included in the precondition of the await.
- $\forall k \in [1..j]\colon b[k] = a[k]$
  By the invariant $\forall k \in [1..j-1]\colon b[k] = a[k]$ holds. We only have to prove that $b[j] = a[j]$. Now remember that $b[j]$ has to be replaced with x. In fact we have to prove that $x = a[j]$. Now consider the following condition included in the invariant:
  $i > j \Rightarrow x = a[i-1]$
  In our case $i > j$ since $i = j+1$, so we can deduce that $x = a[i-1]$ holds. Then $x = a[j]$ also holds, since $i-1 = j$
- $i > j+1 \Rightarrow x = a[i-1]$
  The logical implication is true, because $i > j+1$ is false (we know that $i = j+1$).
- $0 \leq i-1 \leq j+1 \leq i \leq n+1$
  This is the only remaining condition we have to prove.
  - $0 \leq i-1$
    This condition is true, because it is included in the invariant.
  - $i-1 \leq j+1$
    $i-1$ cannot be greater than $j+1$. This holds, since $i-1 = j$ and j is not greater than j.
  - $j+1 \leq i$
    $j+1$ cannot be greater than i. This holds, since $j = i-1$ and i (that is equivalent to $j+1$) is not greater than i.
  - $i+1 \leq n+1$
    This condition is true, because it is included in the invariant.