Tutorials 04: Programming Language Semantics

Exercise 01

Check whether the following statements are theorems:

{x = 3 and y = 2} If x = 3 : If y > x : y := x - 1 Else x := x - 1 EndIf EndIf {x = y}

The statement containes nested "if"s.

Precondition:

$$P \equiv (x=3 \land y=2)$$

Outer "if" condition:

$$B1 \equiv (x = 3)$$

Since P⇒B1, the then-branch well be executed.
There is no else, so execution continues inside the nested if.
Inner condition:

$$B2 \equiv (y > x)$$

Under the precondition $x=3 \land y=2$:

$$y > x \text{ is } 2 > 3 \text{ which is false}$$

Therefore, the else branch is executed:

$$x := x - 1$$

Assignment rule:

$$\{Q[x \leftarrow E]\}\ x := E\ \{Q\}$$

Here:

- Assignment: $x := x - 1$
- Desired postcondition: $Q \equiv (x = y)$

Compute the weakest precondition:

$$Q[x \leftarrow x - 1] \equiv (x - 1 = y)$$

Check whether the state before assignment satisfies it:
Before assignment:

$$x = 3,\ y = 2$$

Evaluate:

$$x - 1 = y \Rightarrow 3 - 1 = 2 \Rightarrow 2 = 2$$

Therefor, this first statement is a theorem.

{x # 3 and x = y} If x = 3 : If y > x : y := x - 1 Else x := x - 1 EndIf EndIf {x = y}

Outer condition:

$$B \equiv (x = 3)$$

Precondition:

$$P \equiv (x \# 3 \land x = y)$$

We have:

$$P \Rightarrow \neg B$$

Therefore, the condition is false, and since there is no else, the command behaves as skip.
Hoare rule for if B then S end:

$$\frac{\{P \wedge B\}\ S\ \{Q\} \qquad P \wedge \neg B \Rightarrow Q}{\{P\}\ \textbf{if}\ B\ \textbf{then}\ S\ \textbf{end}\ \{Q\}}$$

Here:

- P ∧ B is false, so the first premise is trivially satisfied
- We must check:

$$P \wedge \neg B \Rightarrow Q$$

Postcondition:

$$Q \equiv (x = y)$$

From the precondition:

$$x \# 3 \wedge x = y$$

Clearly:

$$x = 3 \wedge x = y \Rightarrow x = y$$

Therefor, this second statement is a theorem.

{x = 3 and x = y} If x = 3 : If y > x : y := x - 1 Else x := x - 1 EndIf EndIf {x = y}

Outer condition:

$$B1 \equiv (x = 3)$$

Precondition:

$$P \equiv (x = 3 \wedge x = y)$$

We have:

$$P \Rightarrow B1$$

Therefore, the then-branch is always executed.
Inner condition:

$$B2 \equiv (y > x)$$

From the precondition x=3 ∧ x=y, we get:

$$y = 3$$

Thus:

$$y > x \equiv 3 > 3 \text{ is false}$$

So the else branch is executed:

$$x := x - 1$$

Initial state:

$$x = 3,\ y = 3$$

After executing:

$$x := x - 1$$

Final state:

$$x = 2,\ y = 3$$

Postcondition:

$$Q \equiv (x = y)$$

But in the final state:

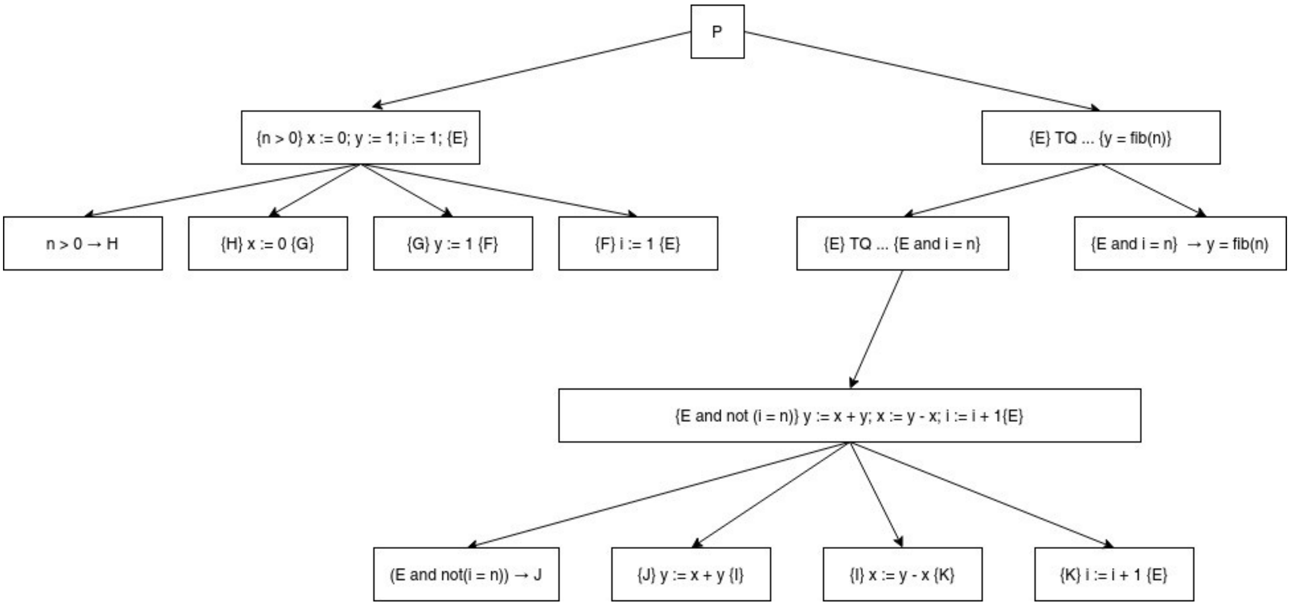
$$2 \# 3$$

Therefor, this third statement is **not** a theorem.

Exercise 02

a) Give the proof tree and show whether the following statement is a theorem:

{n > 0} x:=0; y:=1; i:=1; While i < n : y := x+y; x := y-x; i := i+1 EndWhile {y = Fib(n)}



We can keep the assignments as a block or considered every assignement as a separate instruction as given in the proof tree above.

In the following proof, the loop invariant (E) = $x = Fib(i-1) \land y = Fib(i)$

{ n > 0 }

x := 0;

y := 1;

i := 1;

{ $x = Fib(i-1) \land y = Fib(i) \land i < n$ } ***(E and i < n)***

while i < n do

{ $x = Fib(i-1) \land y = Fib(i) \land i < n$ }

y := x + y;

x := y − x;

i := i + 1

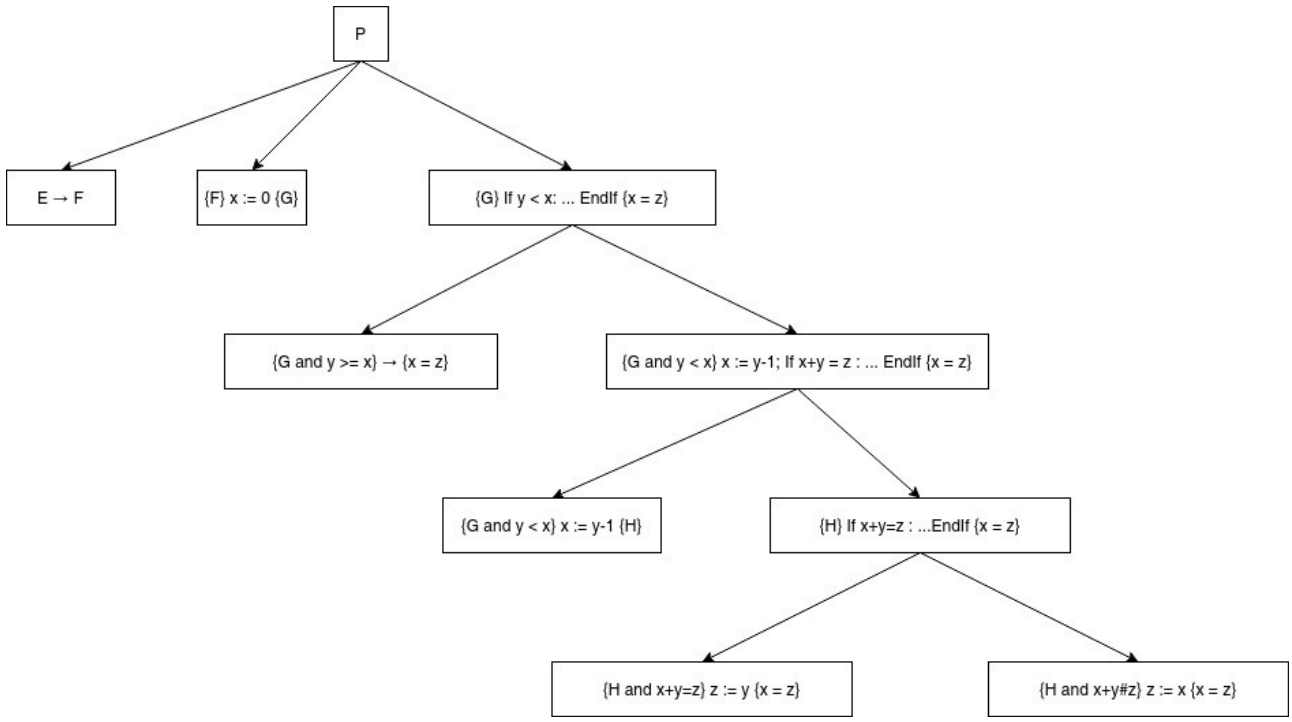
{ $x = Fib(i-1) \land y = Fib(i)$ } ***(E)***

end

{ y = Fib(n) } ***(E and i = n → y = Fib(n))***

b) Give the proof tree and find the weakest precondition for the statement to be a theorem:

{E} x:=0; If y < x : x := y-1; If x+y = z : z := y Else z := x EndIf EndIf {x = z}



- If y >= 0, we must have E → z = x = 0
- If y < 0 Then, we pass to the inner if/else structure

$$\frac{\{P \wedge B\}\ S_1\ \{Q\} \qquad \{P \wedge \neg B\}\ S_2\ \{Q\}}{\{P\}\ \textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{end}\ \{Q\}}$$

  - If z = x + y, we must have E → z = x = y, and that's impossible because x = y – 1, therefor E → z # x + y, so E → z # 2 * y – 1
  - If z # x + y, we must have E → z = x and that's the same as postcondition.

By combining the two branches :

E = (y >= 0 and z = 0) or (y < 0 and z # 2 * y – 1)

Exercise 03

Show whether the statement {n >= 0)} P {x = n*n} is a theorem in Hoare's formal system. P being the following program:

```
i := 0;
j := 0;
x := 0;
While   (j < n)
      x := x+1;
```

```
 i := i+1;
If  (i = n)
     j := j+1;
     i := 0;
EndIf
EndWhile
```

For this loop, the invariant is : **x = j * n + i**

Before the loop x = 0 = 0 * n + 0 (valide)

After the loop, j = n (end of the loop), i = 0 (in the last iteration the condition of the “if” is true), therefor x = n * n + 0 = n * n (the postcondition).

The tree must be built like Q1 from Exercise 2.