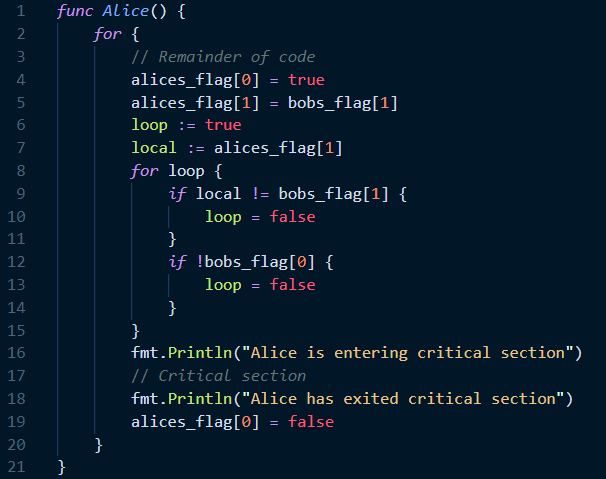
# Implementing await by busy waiting

Demonstrated by Ýmir Þórleifsson & Bjartur Þórhallsson

The code snippet shows the code for Alice after swapping the order of the conditional statements.

We would like to show that, when the statements where switched, the behaviour of the code did not change.

We assume the threads each get their turn to execute instructions, so that starvation is not possible, otherwise we could add a thread.Sleep(0) in the end of the for loop and critical section. To not waste executions on checking other threads, when they should instead be running.

Firstly, what is the function of the loop? It prevents Alice from entering the CS at the same time as Bob, by keeping Alice in the loop if Bob is inside the CS, it also prevents both parties from being stuck in the loop together, by always allowing Alice to proceed in that case.

Secondly, does the functionality of the loop change if we switch the if statements? No. Theses if statements to be independent and

We use weakest precondition calculus to prove that this works:

Wp(if, loop=false) <=> wp(if, R)

((alices\_flag[1] ≠ bobs\_flag[1]) ∨ ¬ bobs\_flag[0]) ∧

(alices\_flag[1] ≠ bobs\_flag[1] => wp(loop=false)) ∧

(¬ bobs\_flag[0] => wp(loop=false))

We simplify this to the following propositional formula:

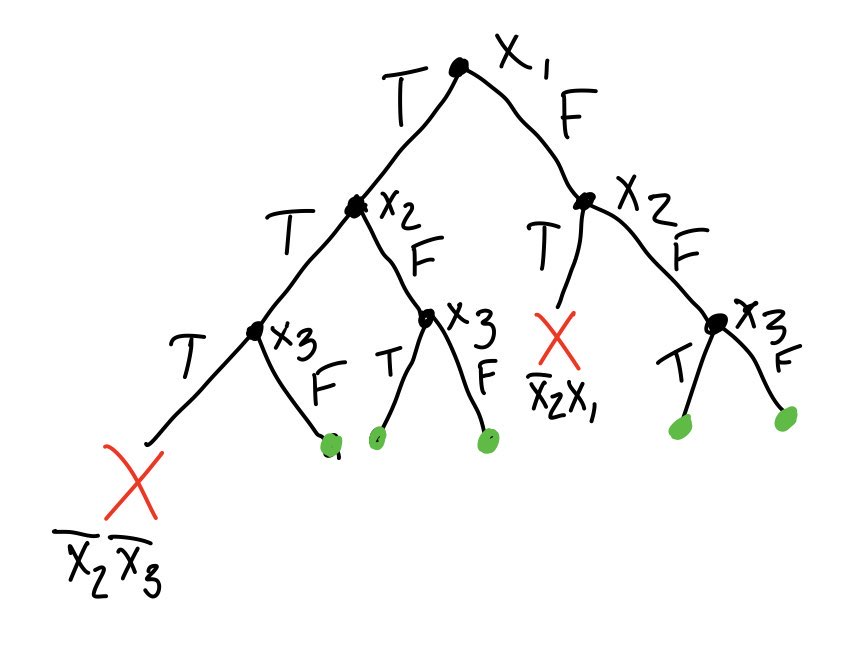
x1 = alices\_flag[1], x2 = bobs\_flag[0], x3 = bobs\_flag[1]

(x1 ∧ ¬ x3) ∨ ¬ x2

Then we transform this into conjunctive normal form (using the distributive law):

(¬ x2 ∨ x1) ∧ (¬ x2 ∨ ¬ x3)

Now we perform DPLL to find the truth values for the x’s

There are only two states that do not satisfy the CNF formula:  
x1 = x2 = x3 = True:  
 This state is impossible to get to because Bob always sets his second flag to the opposite of Alice’s second flag, thus x1 = True and x3 = True is impossible.  
x1 = False and x2 = True:  
x1 being False would mean that bob would raise his second flag (x3) resulting in Alice being able to exit the loop and enter the critical section.

Example:

Bob's dog is in the garden (CS). Alice raises her first and second flag, sits down and waits for either of Bob’s flags to lower, the first flag means that he has let is dog outside, the second flag is the turn. Bob must lower his flag to let his dog inside, signalling to Alice that she can let her dog outside.

We can conclude that swapping the conditional statements has no effect on the postcondition. It still satisfies mutual exclusion, prevents deadlock, and starvation.