Computational Thinking (CO2412): Tutorial – Calendar Week 42

Program Analysis

M. Horsch, O. Kerr, School of Psychology and Computer Science

1.2.1. Loop invariants

Consider the iterative code[1] that computes the *n*th element of the Fibonacci sequence:

```
def fibonacci_iter(n):
    fibo = [0, 1]
    for k in range(2, n+1):
        fibo.append(fibo[k-1] + fibo[k-2])
    return fibo[n]
```

As part of the specification (contract) for this function, it may be assumed that the argument n is a natural number or zero.

a) Draw a program flow graph for this code similar to the ones from the lecture.[2] Include labelled control points representing execution states such as S_1, S_2, *etc.*
b) Describe the conditions at each of the execution states, employing *loop invariants* for states inside the loop body. Use this to prove that the function indeed has the desired outcome, *i.e.*, it returns the *n*th Fibonacci number.

1.2.2. Memory efficiency

1. Explain why we say that the algorithm above has $\mathsf{O}(n)$ memory efficiency.
2. Rewrite the algorithm (and code) such that it has $\mathsf{O}(1)$ memory efficiency.

Submission deadline: 6th November 2021; discussion planned for 18th November 2021. Group work by up to four people is welcome.

[1] See `https://home.bawue.de/~horsch/teaching/co2412/material/iterations-and-recursions.ipynb`.
[2] See `https://home.bawue.de/~horsch/teaching/co2412/slides/CO2412-L1.2-CW42-slides.pdf` and/or p. 6, 36ff. in Nielson *et al.*, *Principles of Program Analysis*, 2nd printing, Springer, **2005**.