Assignment 3

Programming Theory

This assignment is worth 4 points in the final exam. For deadline and instructions for handing in, see the course homepage. Proofs should be presented in exactly the same style as in the compendium (where there is no shortage of examples). It is not necessary to show the steps involving application of commutativity and associativity. You should, however, always state any arithmetic rule you use.

The Bakery Algorithm

\[
\begin{align*}
  a, b, pc_1, pc_2 & := 0, 0, T_1, T_2; \\
  \textbf{do} & \quad pc_1 = T_1 \rightarrow pc_1, a := W_1, b + 1 \\
  \quad & \quad pc_1 = W_1 \land (a < b \lor b = 0) \rightarrow pc_1 := C_1 \\
  \quad & \quad pc_1 = C_1 \rightarrow pc_1, a := T_1, 0 \\
  \quad & \quad pc_2 = T_2 \rightarrow pc_2, b := W_2, a + 1 \\
  \quad & \quad pc_2 = W_2 \land (b < a \lor a = 0) \rightarrow pc_2 := C_2 \\
  \quad & \quad pc_2 = C_2 \rightarrow pc_2, b := T_2, 0 \\
  \textbf{od}
\end{align*}
\]

Prove that the implementation is correct, in the sense that the program respects the mutual exclusion property.

Hints:

- Start by getting a feeling for what the algorithm does. Simulate it using pen and paper.
- Respecting the mutual exclusion property means that \( \neg((pc_1 = C_1) \land (pc_2 = C_2)) \) is an invariant. However, proving this directly is not feasible, so you have to take a slight detour. You instead have to
  1. find a stronger statement \( P \) that you believe to be an invariant
  2. prove that this \( P \) in fact is an invariant
  3. prove that this \( P \) implies mutual exclusion.
- Finding a good invariant is the hard part here. A good understanding of the algorithm is crucial, but often not enough. The trick therefore is to develop the invariant iteratively: start with some good guess, try to prove that it is an invariant, and then try to see what you lack in order to complete the proof. Now go back and strengthen the invariant with exactly this information, if possible, and start over.