Solutions for CP Exercises, September 18

1. Solution for CP Exam December 1998, Problem 1

Question 1.1

(a) At entry in cs_1 , $C_1 = 1$ due to the loop condition. In cs_1, C_1 is not changed by P_1. Further, the only assignment to C_1 in P_2 is f_2 which cannot change $C_1 = 1$. Thus, $C_1 = 1$ in cs_1.

(b) *Omitted.*

(c) $I \triangleq C_1 = C_2 \Rightarrow C_1 = 0 \land C_2 = 0$

I holds initially since $C_1 = 0 \land C_2 = 0$.

The potenially dangerous actions are changes of C_1 and C_2. For P_1 these are:

a_1: After the execution, C_1 and C_2 are different (cf. (b)). Thus I holds.

d_1: If $C_2 \neq 0$ before the execution, C_1 and C_2 are different afterwards. If $C_2 = 0$ before, both are 0 afterwards. In both cases, I holds.

f_1: Before the execution, $C_1 = 0$ cf. H_1. Since C_1 is not changed by this action, C_1 and C_2 differ after the execution, hence I holds.

By symmetry, we see that I is also preserved by all actions in P_2. Together with I holding initially, we conclude that I is an invariant for the program.

(d) Assume that mutual exclusion was violated: $in\ ks_1 \land in\ ks_2$

According to (a), we should then have $C_1 = 1 \land C_2 = 1$ contradicting I. Thus, mutual exlusion is ensured for this program.

Question 1.2 *(Not required)*

It is assumed that P_1 stays in w_1 for ever:

1. If P_1 stays in w_1 for ever, it follows from H_1 that $C_1 > 0$ must hold forever. Thus, we can assume $\Box in\ w_1$ og $\Box C_1 > 0$ in the following.
2. Due to fair process execution, P_2 will eventually reach e_2 unless it gets stuck somewhere else. Since we assume that it cannot remain in cs_2, it can only get stuck in nc_2 or w_2.
3. When P_2 executes the test in the **if**-statement, we either have $C_1 = 1$ or $C_1 > 1$ (since $\Box C_1 > 0$). In the latter case, P_2 moves to f_2.
4. Due to fair process execution, f_2 will be executed eventually, ant right after this, $C_1 = 1$.
5. If at any time $C_1 = 1$ it will remain so for ever since P_1 is assumed to stay in w_1 and P_2 cannot change C_1 to any other value than 1.
6. When P_2 is in nc_2 we cannot have $C_1 > 1$ according to K_1. Since $\Box C_1 > 0$, it must therefore be 1. Thus, we have $\Box in\ ik_2 \Rightarrow \Box C_1 = 1$.
7. If $\Box C_1 = 1$, P_1 will eventually discover this and leave w_1

8. This contradics the assumption $\Box in\ w_1$.

Question 1.3

(a) Since G_i and H_i now become *local invariants*, these are immediately seen to still hold. Likewise, the argument for I holds and hence the mutual exclusion proof is still valid.

(b) Consider the following program execution:

	C_1	C_2
Initielt	0	0
P_1 executes nc_1, a_1, and enters cs_1	1	0
P_2 executes nc_2, a_2	1	2
P_1 executes d_1, nc_1, a_1	3	2

Both process are now caught in w_1. Since a deadlock can occur between the entry-protocols, the implementation is no longer resolute.