# G52CON: Exercise 6, CTL

Consider the concurrent program consisting of the two processes shown below:

```
           boolean turn = r1 = r2 = false
process 1                         process 2

while(true) {                   while(true) {
   r1 = turn;                      r2 = turn;
   if (!r1) {                      if (r2) {
      <crit1>;                        <crit2>;
      turn = true;                    turn = false;
   }                               }
}                               }
```

States are described by the following propositional variables:

- $t$ for `turn` equals true;
- $r_1$ for `r1` equals true;
- $r_2$ for `r2` equals true;
- $c_1$ for process 1 is in its critical section;
- $c_2$ for process 2 is in its critical section;

The starting state is $s_0 = (\neg t, \neg r_1, \neg r_2, \neg c_1, \neg c_2)$. Each process performs the following sequence of actions: copy the value of variable `turn` into its local variable `r1` or `r2`; evaluate the if statement; depending on the outcome, either do its critical section and set turn to be true (false), or return to the first step. Actions can be arbitrarily interleaved. As usual we assume that assignment is an atomic operation. If we are in a state where for example $r_1$ holds and $t$ does not and process 1 executes atomic action `r1 = turn` we make a transition to a state where $\neg r_1$ holds. In this example we can also assume that evaluating the if statement is an atomic operation since the value of the local variable can't be changed by another process. So if we are in the state where $\neg r_1$ holds and process 1 evaluates the condition of its if statement, we move to a state where $c_1$ is true.

## Questions:

(a). Draw the state transition diagram.

(b). Write a CTL formula expressing the following property: *it is impossible that* `r1` *is true and process 1 is in its critical section.* Is this formula true at the start state of the transition system you drew? Justify your answer using the truth definition for CTL formulas.

(c). Write a CTL formula expressing the property of non-strict scheduling: *process 1 and process 2 don't always take turns at entering critical section, but instead for example process 1 can go into critical section several times in a row.* (Hint: you may need to use Until operator.) Is this formula true at the start state of the transition system you drew? Justify your answer using the truth definition for CTL formulas.

**Answer:**

(a). The following states are possible:

$$s_0 = (\neg t, \neg r_1, \neg r_2, \neg c_1, \neg c_2)$$
$$s_1 = (\neg t, \neg r_1, \neg r_2, c_1, \neg c_2)$$
$$s_2 = (t, \neg r_1, \neg r_2, \neg c_1, \neg c_2)$$
$$s_3 = (t, r_1, \neg r_2, \neg c_1, \neg c_2)$$
$$s_4 = (t, \neg r_1, r_2, \neg c_1, \neg c_2);$$
$$s_5 = (t, r_1, r_2, \neg c_1, \neg c_2);$$
$$s_6 = (t, \neg r_1, r_2, \neg c_1, c_2);$$
$$s_7 = (t, r_1, r_2, \neg c_1, c_2);$$
$$s_8 = (\neg t, r_1, r_2, \neg c_1, \neg c_2);$$
$$s_9 = (\neg t, \neg r_1, r_2, \neg c_1, \neg c_2);$$
$$s_{10} = (\neg t, r_1, \neg r_2, \neg c_1, \neg c_2);$$
$$s_{11} = (\neg t, \neg r_1, r_2, c_1, \neg c_2);$$

Transitions:

$(s_0, s_0)$ by process 1 executing first assignment; by process 2 executing first assignment; by process 2 evaluating if.

$(s_0, s_1)$ by process 1 evaluating if.

$(s_1, s_2)$ by process 1 executing second assignment.

$(s_1, s_1)$ by process 2 evaluating if.

$(s_2, s_3)$ by process process 1 executing first assignment.

$(s_2, s_4)$ by process process 2 executing first assignment.

$(s_3, s_5)$ by process process 2 executing first assignment.

$(s_3, s_3)$ by process 1 evaluating if.

$(s_4, s_6)$ by process 2 evaluating if.

$(s_4, s_5)$ by process process 1 executing first assignment.

$(s_5, s_7)$ by process 2 evaluating if.

$(s_5, s_5)$ by process 1 evaluating if, executing first assignment.

$(s_6, s_9)$ by process 2 executing second assignment.

$(s_6, s_7)$ by process 1 executing first assignment.

$(s_7, s_8)$ by process 2 executing second assignment.

$(s_7, s_7)$ by process 1 evaluating if, executing first assignment.

$(s_8, s_9)$ by process 1 executing first assignment.

$(s_8, s_{10})$ by process 2 executing first assignment.

$(s_9, s_{11})$ by process 1 evaluating if.

$(s_9, s_0)$ by process 2 executing first assignment.

$(s_{10}, s_0)$ by process 1 executing first assignment.

$(s_{10}, s_{11})$ by process 1 evaluating if.

$(s_{11}, s_4)$ by process 1 executing second assignment.

$(s_{11}, s_1)$ by process 2 executing first assignment.

(b). $AG(\neg(r_1 \wedge c_1))$. This is true since all states satisfy $\neg(r_1 \wedge c_1)$.

(c). $EF(c_1 \wedge EXE(\neg c_2 U c_1))$ (there exists a path such that $c_1$ is true at some point on that path and later (not at the same point) $c_1$ is true again and in between $c_2$ is not true). The statement is false: there is no path where critical sections don't alternate.