Task 1:

The following state diagram describes a microwave oven, which has one button and a light bulb inside. If the door is closed and the button is pressed once, the oven starts cooking for 1 minute. Cooking time can be extended by pressing the button while cooking. The oven is controlled by a timer. Cooking is done using the power tube. If the door is opened, cooking is interrupted. Initially, the door is closed and the oven is not cooking.

Note: All labels on transitions refer to events while actions are denoted inside states.

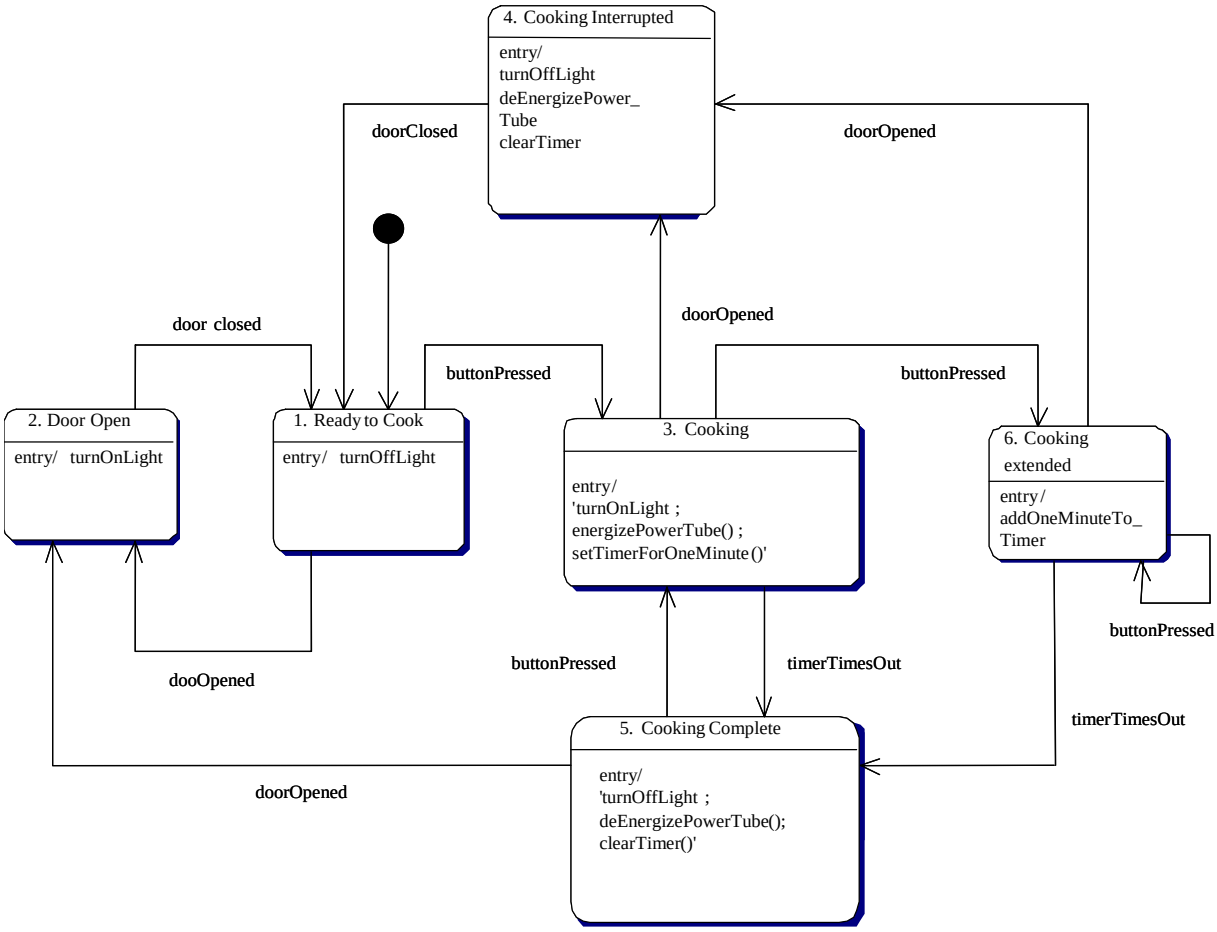

Figure 1: Statechart for a microwave oven

Questions:

a) Is it possible to open the door while the oven is cooking and receive radiation from the power tube (i.e. the power tube was not de-energized)? Justify your answer in terms of the states and transitions shown in Figure 1.

Answer: No, because the only states where the power tube is energized are Cooking and Cooking Extended (no need to explain why this is so). In both cases, whenever the door is opened (reflected by the event doorOpened) the next state is Cooking Interrupted, upon entry to this event the action deEnergizePowerTube will be executed.

b) What happens if you press the button while the door is open? Justify your answer in terms of the states and transitions shown in Figure 1.

Answer: Nothing. The only states where the door is open are Door Open and Cooking interrupted. Given that no transition labelled with the event doorOpened leaves these states, pressing the button does not change the state of the microwave oven.

c) Assume that the oven is cooking and you open the door before the cooking time is over (i.e. you interrupt cooking). What happens if you immediately close the door? Does the oven resume cooking to complete the preset time? Justify your answer in terms of the states and transitions shown in Figure 1.

Answer: No.
Upon interrupting cooking, the timer is cleared (i.e. set to zero). Closing the door causes the microwave oven to leave the state Cooking interrupted and arrive to Ready to cook, where the cooking is not resumed.

Aufgabe Bahnübergang

The events of interest are:

- Train approaches on Track A
- Train leaves on Track A
- Train approaches on Track B
- Train leaves on Track B

The start state should be Green. The student may assume that after a train approaches on a track, no other train will approach on the same track (under normal operating conditions). Therefore, no counter is needed for the number of incoming trains that have not left the critical area. Later we'll relax this assumption, but one (valid) answer looks as follows (Figure 2):

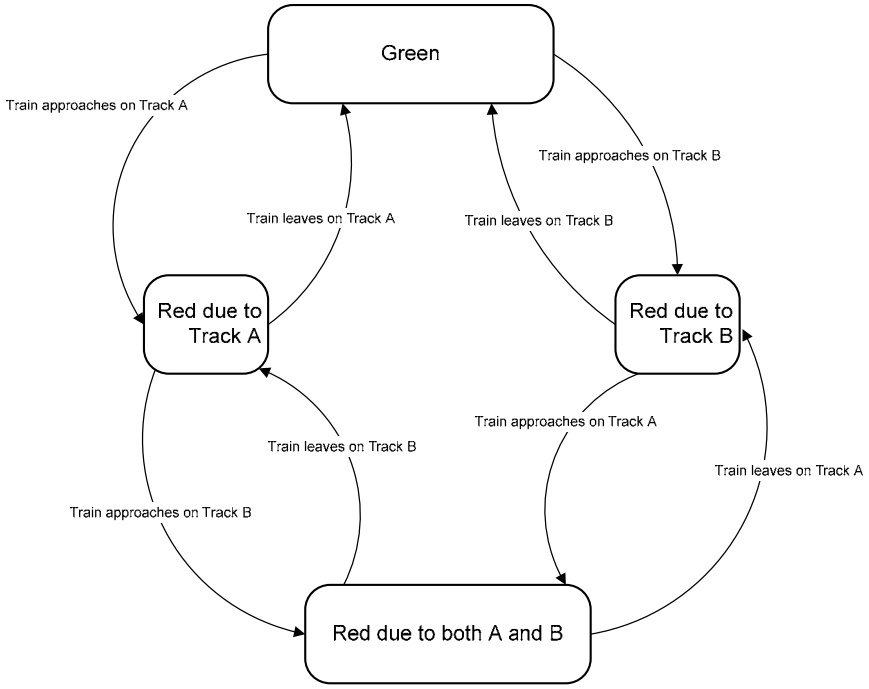

Figure 2

Considering real world conditions, timeouts are unavoidable to account for situations where a “train leaves” signal is lost, as well as conditions for other failure modes of equipment. However, students are not expected to consider any of those eventualities: equipment is assumed to be perfect.

The student may suggest a statemachine which can deal with multiple occurrences of “train approaching” events on the same track before one or more “train leaves” signals on that track are received. The corresponding statechart will make use of two statemachine-wide counters (one for each Track) with the number of “train approaching” events not yet matched by “train leaves” events (see Figure 3). The state where the light is red could be partitioned into two *parallel regions* (one for each track), but for the purposes of this example the statechart is understandable anyway.

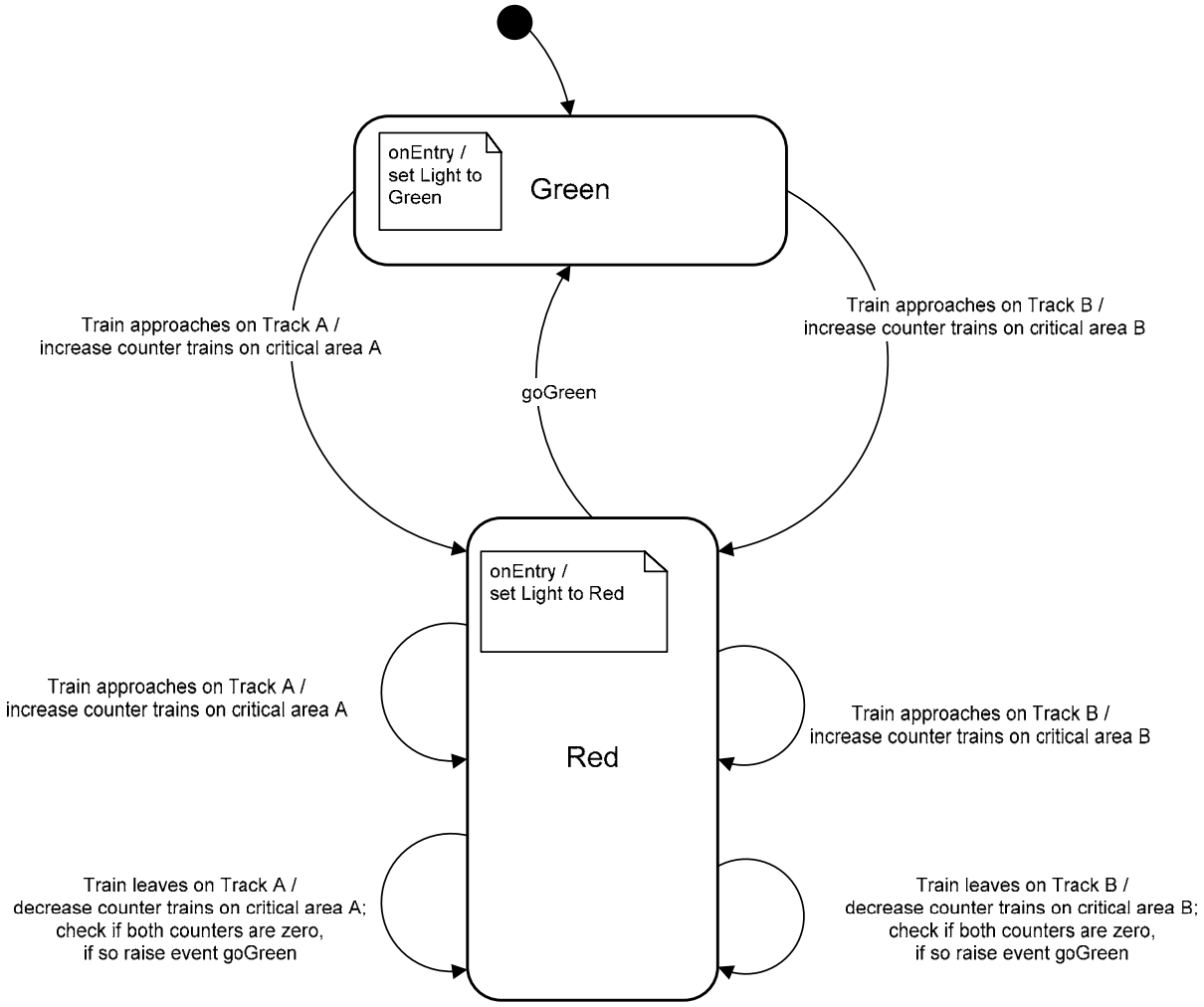
onEntry /
set Light to
Green
Green
Train approaches on Track A /
increase counter trains on critical area A
goGreen
Train approaches on Track B /
increase counter trains on critical area B
onEntry /
set Light to Red
Train approaches on Track A /
increase counter trains on critical area A
Red
Train approaches on Track B /
increase counter trains on critical area B
Train leaves on Track A /
decrease counter trains on critical area A;
check if both counters are zero,
if so raise event goGreen
Train leaves on Track B /
decrease counter trains on critical area B;
check if both counters are zero,
if so raise event goGreen

Figure 3