DIT636/DAT560 - Finite State Verification Exercise

If you have not finished the activity from Lecture 14 (Finite State Verification), do so!

Temporal Operators: A quick reference list. p is a Boolean predicate or atomic variable.

- G p: p holds globally at every state on the path from now until the end
- F p: p holds at some future state on the path (but not all future states)
- X p: p holds at the next state on the path
- p U q: q holds at some state on the path and p holds at every state before the first state at which q holds.
- A: for all paths reaching out from a state, used in CTL as a modifier for the above properties (AG p)
- E: for one or more paths reaching out from a state (but not all), used in CTL as a modifier for the above properties (EF p)

An LTL example:

- G (MESSAGE_SENT -> F (MESSAGE_RECEIVED))
- It is always true (G), that if the message is sent (property MESSAGE_SENT is true), then at some point after it is sent (F), the message will be received (property MESSAGE_RECEIVED will become true).

A CTL example:

- EG (WIND -> AF (RAIN))
- There is a potential future where it is a certainty (EG) that - if there is wind (property WIND is true) - it will always be followed eventually (AF) by rain (property RAIN will become true).

You may find the following resources helpful in completing the activity:

- http://nusmv.fbk.eu/ - NuSMV homepage (tool download, tutorials, etc.)
- Lecture 14 slides (model and property examples)

Consider a simple microwave controller modeled as a finite state machine using the following state variables:

- Door: {Open, Closed} -- sensor input indicating state of the door
- Button: {None, Start, Stop} -- button press (assumes at most one at a time)
- Timer: 0...999 -- (remaining) seconds to cook
- Cooking: Boolean -- state of the heating element

A partial model is presented below:

```
MODULE microwave
VAR
    Door: {Open, Closed};
    Button: {None, Start, Stop};
    Timer: 0..999;
    Cooking: boolean;
ASSIGN
    init(Door) := Closed;
    init(Button) := None;
    init(Timer) := 0;
    next(Timer) :=
    case
        Timer > 0 & Cooking=TRUE : Timer - 1;
        Timer > 0 & Cooking=FALSE & Button!=Stop : Timer;
        Button=Stop : 0;
        Timer=0 : 0..999;
        TRUE: Timer;
    esac;
    init(Cooking) := FALSE;
    next(Cooking) :=
    case
        -- Suggestion: Start by defining the conditions that would cause
        -- cooking to start. Then add conditions that would make it stop.
        -- Finally, ensure it will continue running if it is supposed to.
        (YOU MUST FILL THIS IN)
        TRUE: FALSE;
    esac;

SPEC AG (Door = Open -> AX(!Cooking));
LTLSPEC G (Cooking -> F (!Cooking));
```

1. Complete the model by defining the **`next(Cooking)`** state transitions.
2. For each of the following informal requirements, formulate a temporal logic representation in CTL (the first is given):
    a. The microwave shall stop cooking after the door is opened.
    `AG (Door = Open -> AX (!Cooking))`
    b. The microwave shall cook only as long as there is remaining cooking time.

    c. If the stop button is pressed when the microwave is not cooking, the remaining cook time shall be cleared.

3. For each of the following informal requirements, formulate a temporal logic representation in LTL (the first is given)
    a. It shall never be the case that the microwave can continue cooking indefinitely.
    `G (Cooking -> F (!Cooking))`
    b. If the microwave is cooking, the stop button is not pressed, and the door is not opened, the timer will eventually reach 0.

    c. If the microwave is not currently cooking, and starts cooking in the next state, then the start button must have been pressed while the door is closed and the remaining cook time is set to be greater than one second.