



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

# Exercise 6: Finite State Verification

Gregory Gay  
DIT635 - March 6, 2020



# Finish In-Class Activity First!

# Microwave

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

# Partial Model

```
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.  
(FILL THIS IN)
```

```
TRUE: FALSE;
```

```
esac;
```

# Example Properties

- CTL: The microwave shall stop cooking after the door is opened.
  - $AG (\text{Door} = \text{Open} \rightarrow AX (\neg \text{Cooking}))$
- LTL: It shall never be the case that the microwave can continue cooking indefinitely.
  - $G (\text{Cooking} \rightarrow F (\neg \text{Cooking}))$
- Formulate the other informal requirements in temporal logic.

# Linear Time Logic Formulae

Formulae written with propositional variables (boolean properties), logical operators (and, or, not, implication), and a set of modal operators:

hunger = “I am hungry”

burger = “I eat a burger”

<b>X (next)</b>	X hunger	In the next state, I will be hungry.
<b>G (globally)</b>	G hunger	In all future states, I will be hungry.
<b>F (finally)</b>	F hunger	Eventually, there will be a state where I am hungry.
<b>U (until)</b>	hunger U burger	I will be hungry until I start to eat a burger. (hunger does not need to be true once burger becomes true)
<b>R (release)</b>	hunger R burger	I will cease to be hungry after I eat a burger. (hunger and burger are true at the same time for at least one state before hunger becomes false)

# Computation Tree Logic Formulae

Combines all-path quantifiers with path-specific quantifiers:

<b>A (all)</b>	A hunger	Starting from the current state, I must be hungry on all paths.
<b>E (exists)</b>	E hunger	There must be some path, starting from the current state, where I am hungry.

<b>X (next)</b>	X hunger	In the next state on this path, I will be hungry.
<b>G (globally)</b>	G hunger	In all future states on this path, I will be hungry.
<b>F (finally)</b>	F hunger	Eventually on this path, there will be a state where I am hungry.
<b>U (until)</b>	hunger U burger	On this path, I will be hungry until I start to eat a burger. (I must eventually eat a burger)
<b>W (weak until)</b>	hunger W burger	On this path, I will be hungry until I start to eat a burger. (There is no guarantee that I eat a burger)

# Try to Verify the Model and Properties

- <http://nusmv.fbk.eu/>
  - NuSMV homepage (tool download, tutorials, etc.)
  - Use NuSMV 2.6.
- Try to define next(Cooking) such that the two example properties hold. See if your properties hold.
  - If they don't, make sure the properties are correct.
  - Then, make sure the model is complete and correct.





UNIVERSITY OF  
GOTHENBURG

---



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY