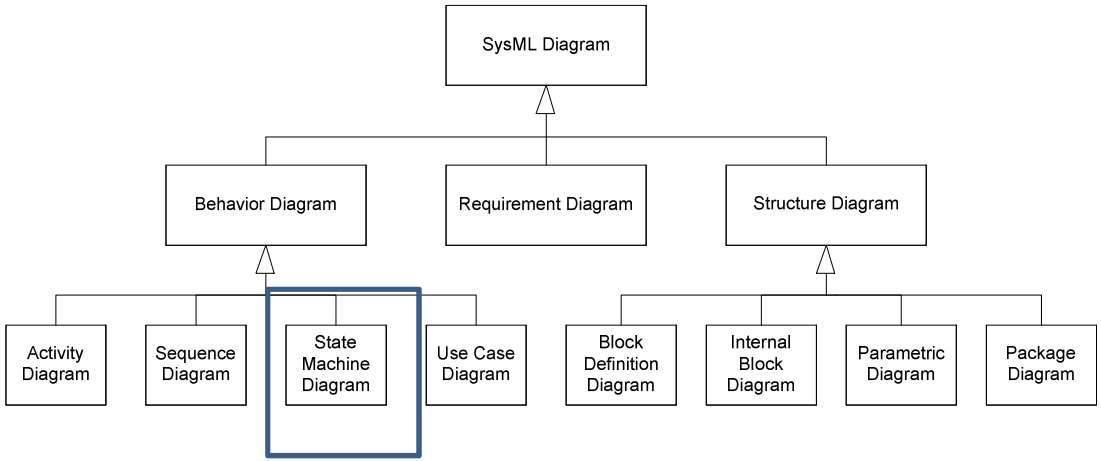
SysML Behavioural Diagrams

State Machine Diagrams

Introduction to Systems Engineering
I2ISE

Introduction



State Machines

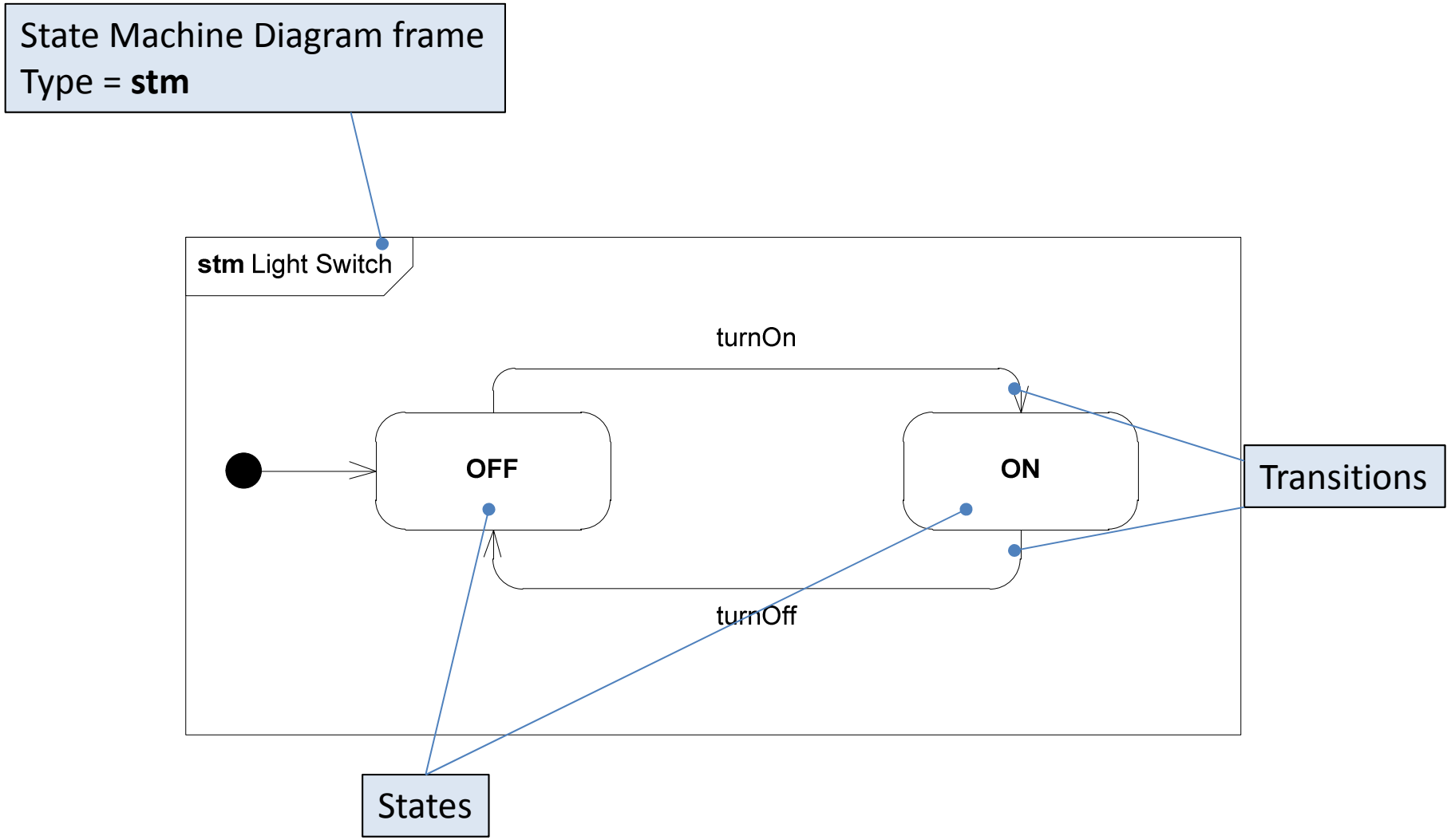
- State Machines Diagrams (**stm**), aka *state charts*, are used to model *state-dependen*t behaviour of a block throughout its lifecycle

- A *state* is some significant condition in the life of a block
  - Typically, different states respond differently to same events

- A *state machine* is always in a certain *state* and will remain there until some *event* causes it to *transition* to another state.

- Any examples?

States and transitions - basics



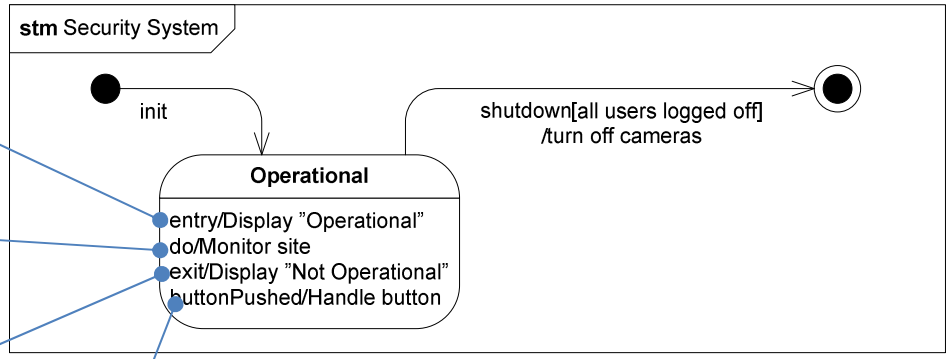
States in detail

entry behaviour is executed on entry into the state

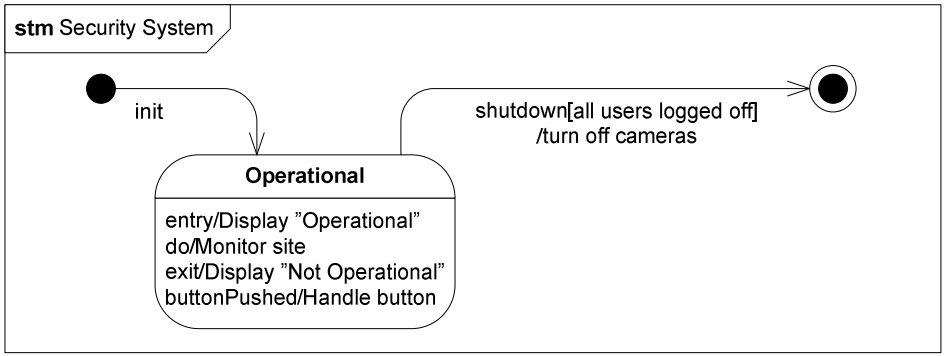
do behaviour is continuously executed after entry until exit

exit behaviour is executed just prior to exit of the state

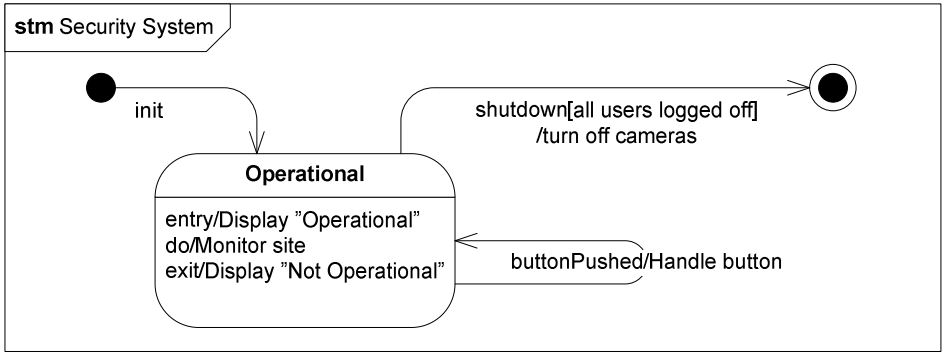
When *buttonPushed* occurs, do behaviour is interrupted and Handle button is executed. Then, do is resumed



States in detail – what's the difference?



buttonPushed →
1. *Handle button*

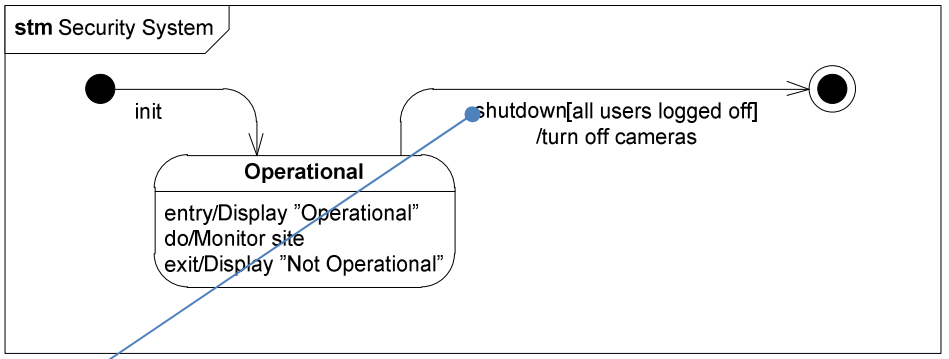


buttonPushed →
1. *Display "Not operational"*
2. *Handle button*
3. *Display "Operational"*

Transitions in detail

- Transitions consist of *trigger*, *guard* and *effect*: `trigger[guard]/effect`

- When `trigger` occurs, `guard` is evaluated.
  - If `guard` is true, `effect` occurs.
  - If not, `trigger` is consumed without effect



Trigger = shutdown
Guard = all users logged off
Effect = turn off cameras

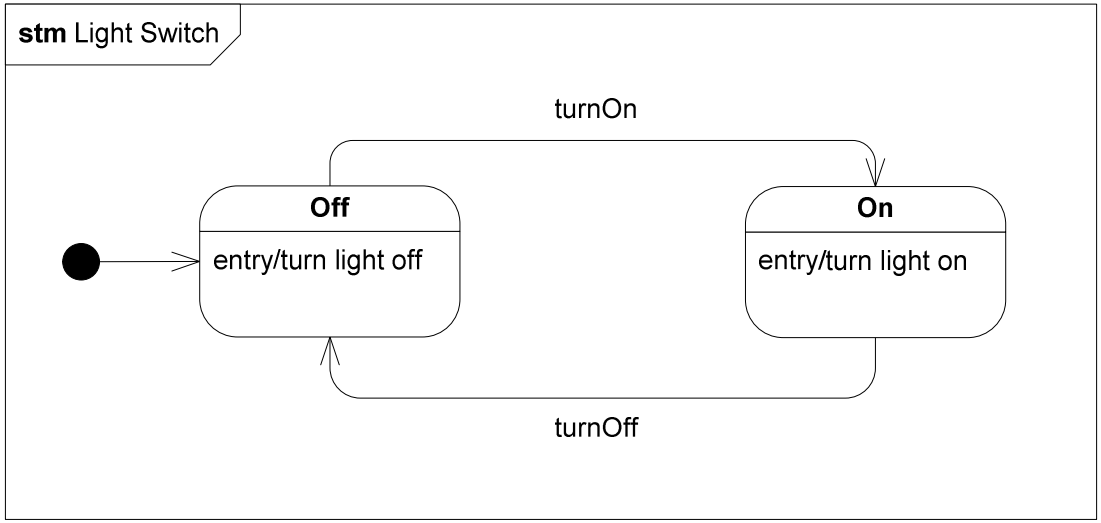
What happens if some user is still logged on?

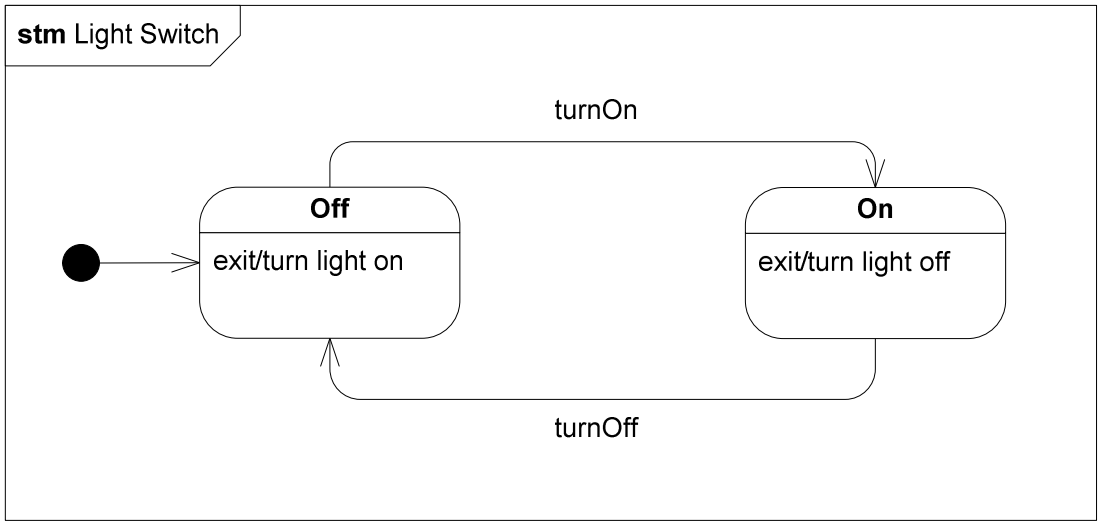
Exercise: Light switch

- Create a state machine diagram for a light switch which controls a light bulb
  - When the light switch is turned On, the light shall be turned on
  - When the light switch is turned Off, the light shall be turned off

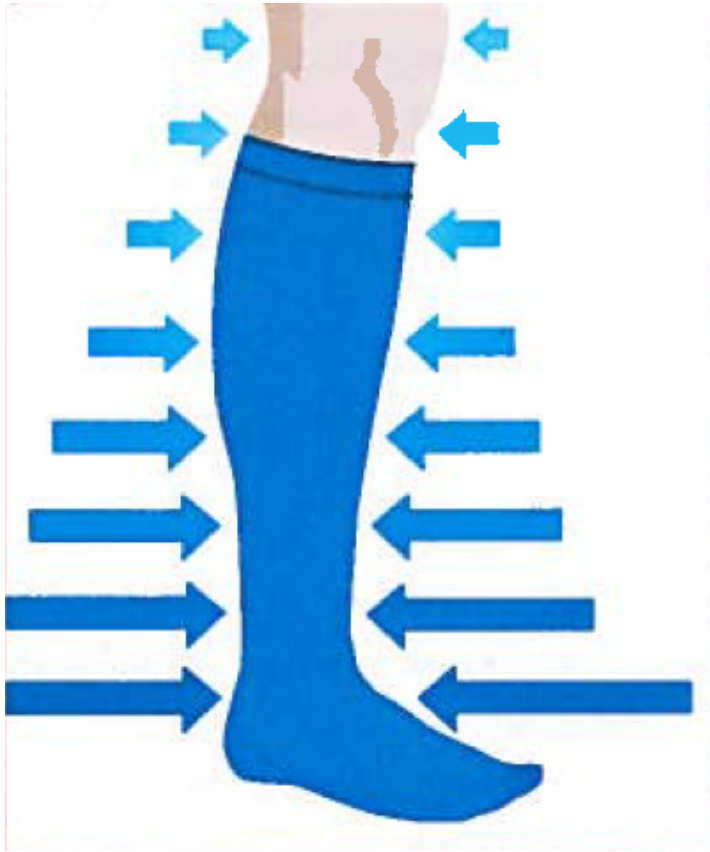
- Your solution should use entry and/or exit actions, not actions on transitions

Exercise: Light switch





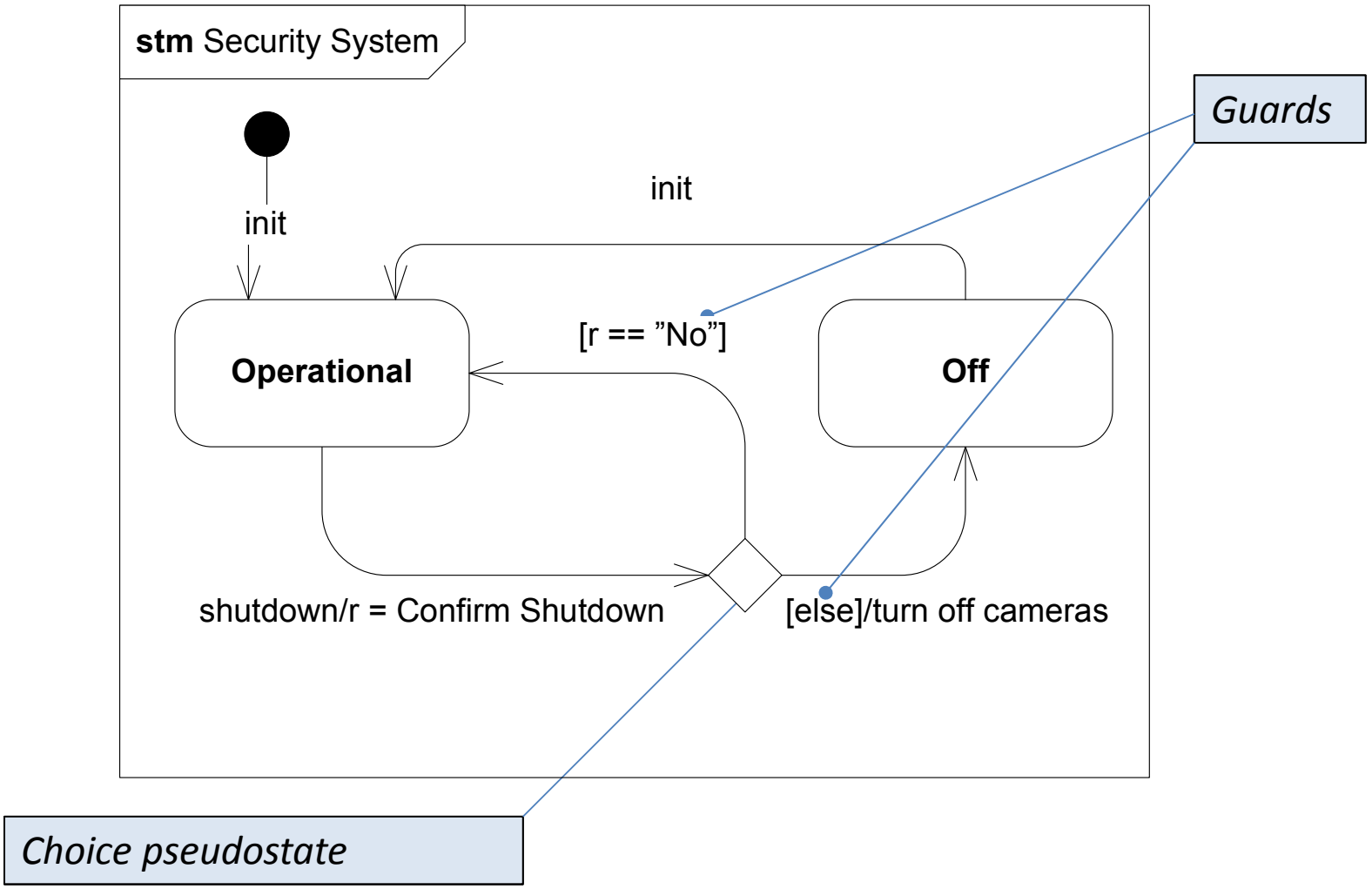
Exercise: Compression stocking 1



Exercise: Compression stocking 1

- Create a state machine diagram for a compression stocking
  - When the RED button is pushed, the stocking compresses.
  - When the GREEN button is pushed, the stocking decompresses.
  - Each second the battery level is checked. If it falls below 2.8V, the stocking decompresses and enters a FAIL SAFE state

Choice pseudostate

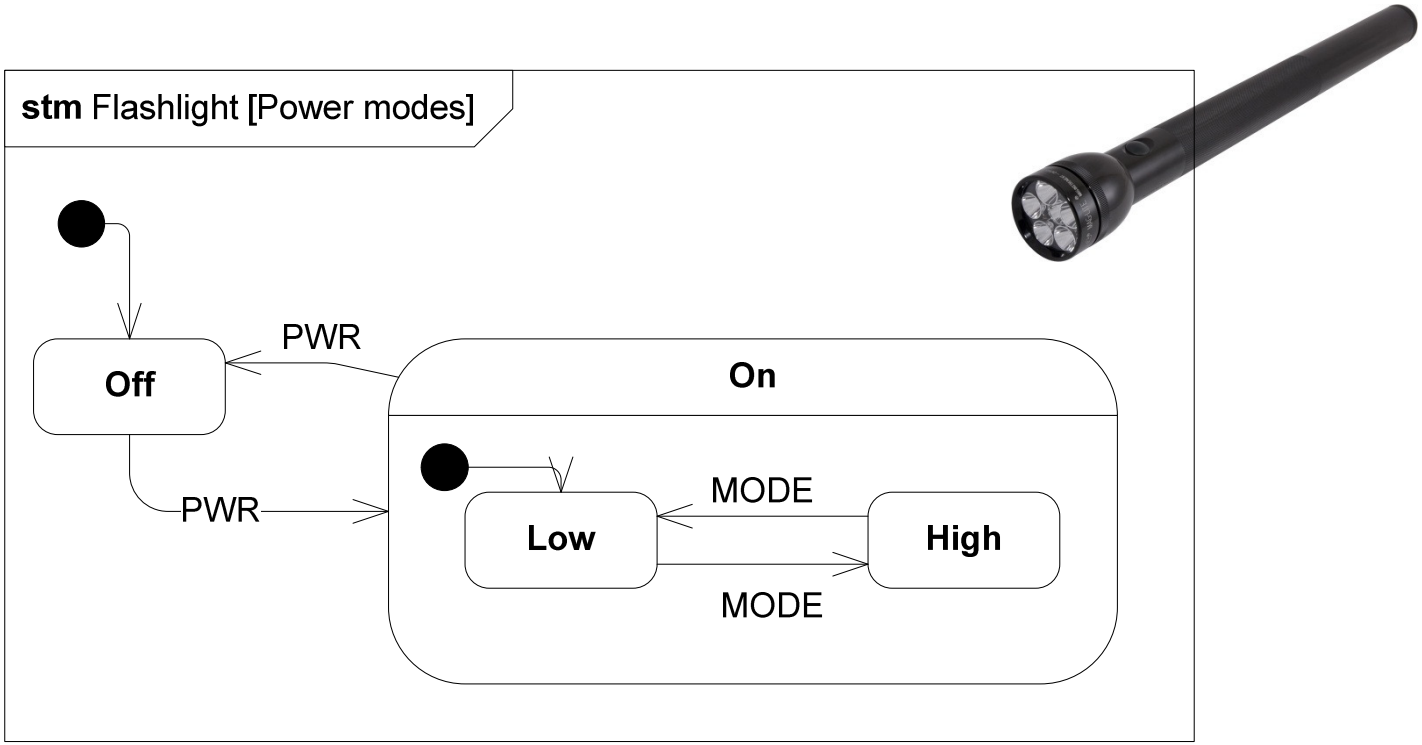


Exercise: Egg Timer 2000

- Create a state machine diagram for an egg timer:
  - The egg timer has four buttons: **`MIN`**, **`SEC`**, **`START`**, **`STOP`**
    - **`MIN`**, **`SEC`**: Increase time by 60 seconds and 1 second, respectively
    - **`START`**: Start countdown
    - **`STOP`**: If running: Stop countdown. If stopped: Clear time. If alarming: Stop alarm
  - Each second, there must be a *tick* event. If ET2000 is running, the remaining number of seconds shall be counted down by 1. If the timer expires, an alarm shall sound.
  - Ignore display updates etc. and concentrate on the setting, counting down and alarming.

States and substates (nested states)

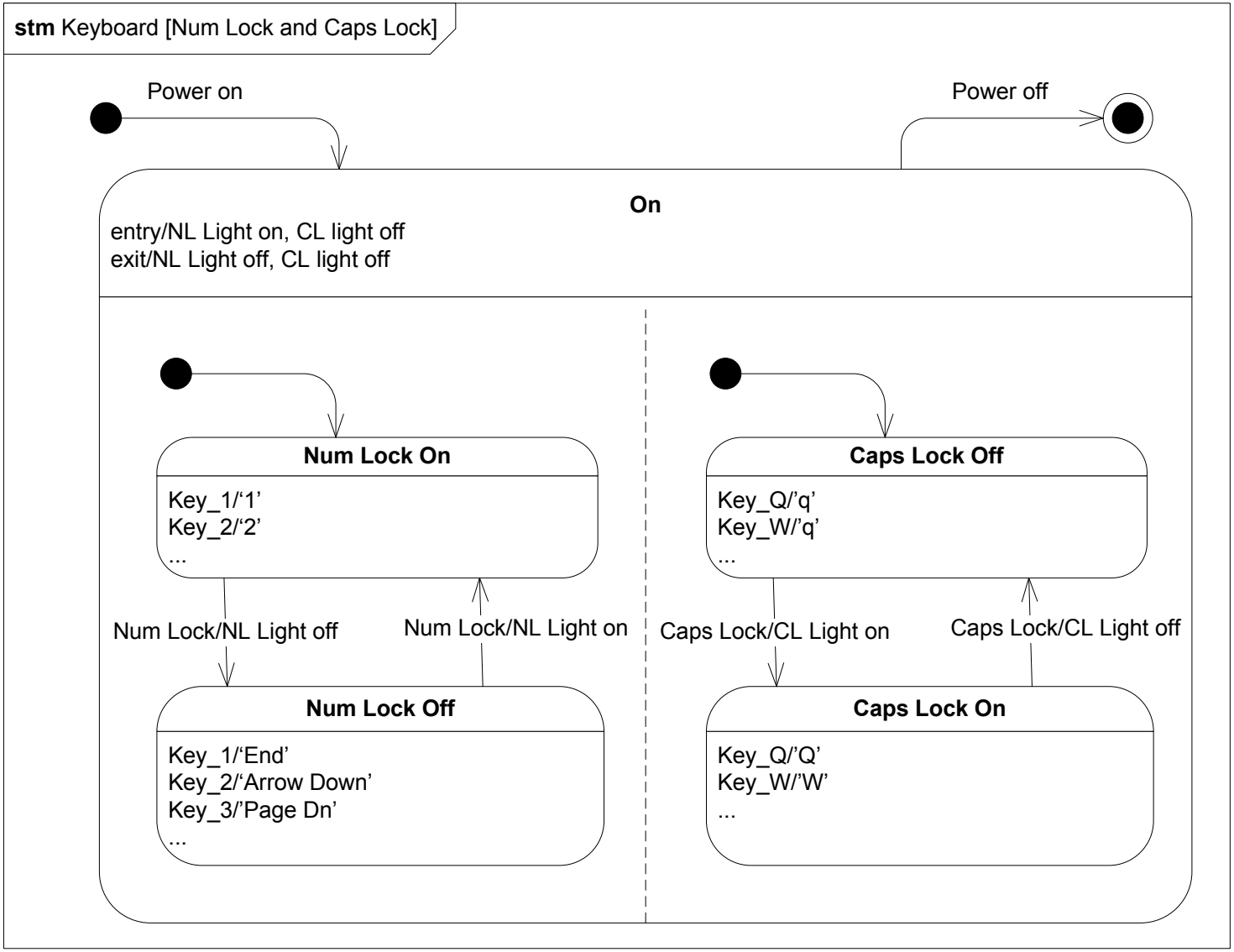
- A state may have substates.
- Example: Flashlight with **`PWR`** and **`MODE`** buttons



States with multiple regions

- A state may have multiple regions (aka. *orthogonal* or *independent substates*)
- If the enclosing state is active, each region will have exactly 1 active state
- State transitions in one region does not affect states in another region.
- State transitions can never transition the boundary between regions

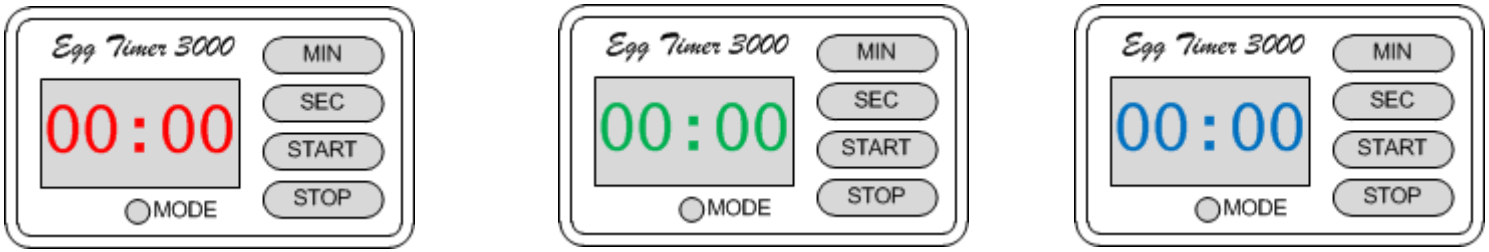
States with multiple regions: Example



Exercise 2 : Pimped Egg Timer 3000

- PET3000 is like ET2000, but the display can be backlit with either red, green or blue light. This is controlled with the **MODE** button which toggles the light.

- Draw it's state machine diagram



Home Exercises

- SysML State Machines (konsol).pdf
- SysML State Machines (telefon).pdf