COMPUTER ENGINEERING

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

EMBEDDED SYSTEM DESIGN

Discrete Dynamics

Doan Duy, Ph. D.

Email: duyd@uit.edu.vn

Objectives

- Understanding basic concepts of Discrete systems
- Able to design an FSM

Contents

1. Introduction to Discrete Dynamics
2. Finite State Machine
3. Language and Framework to design Discrete Systems

Introduction to Discrete Dynamics

Definition

Discrete = “individually separate / distinct”

A **discrete system** is one that operates in a sequence of discrete *steps* or has signals taking discrete *values*.

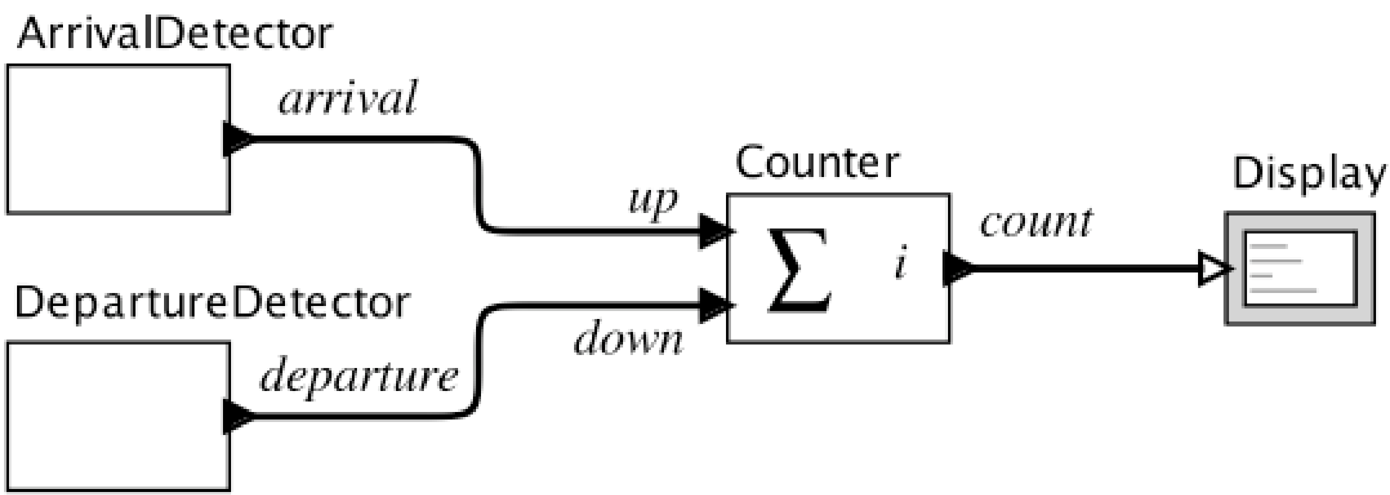
It is said to have **discrete dynamics**.

Example Design Problem

Count the number of cars that are present in a parking garage by sensing cars enter and leave the garage. Show this count on a display.

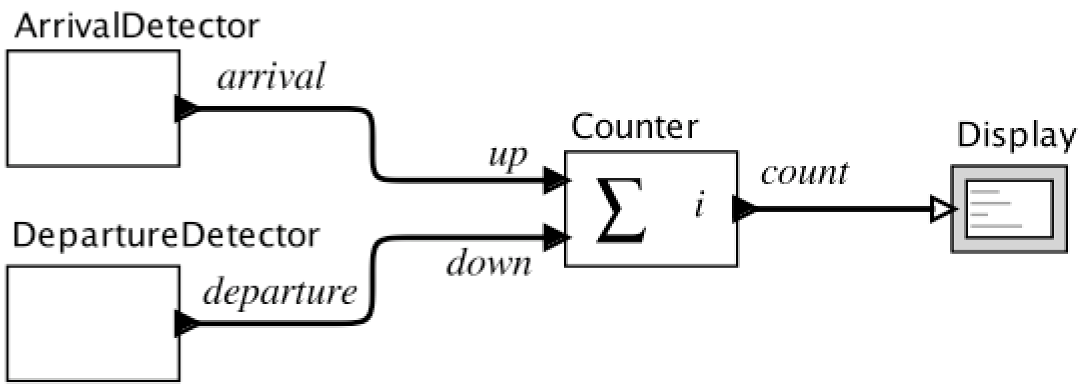
Discrete System

Example: count the number of cars in a parking garage by sensing those that enter and leave:



Discrete System

- Example: count the number of cars that enter and leave a parking garage:



- Pure signal: $up\colon \mathbb{R} \to \{absent, present\}$
- Discrete actor:

$$Counter\colon (\mathbb{R} \to \{absent, present\})^P \to (\mathbb{R} \to \{absent\} \cup \mathbb{N})$$

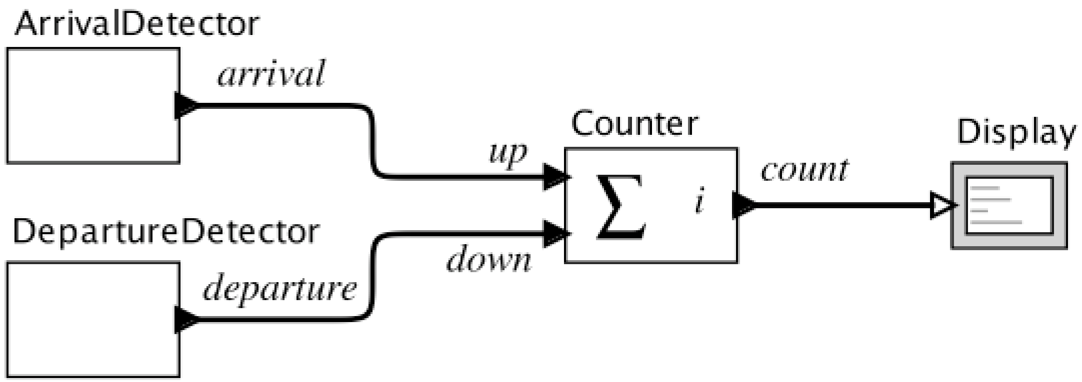
$$P = \{up, down\}$$

Reaction/Transition

For any $t \in \mathbb{R}$ where $up(t) \neq absent$ or *down*$(t) \neq absent$ the Counter **reacts**. It produces an output value in $\mathbb{N}$ and changes its internal **state**.

State: condition of the system at a particular point in time

- Encodes everything about the past that influences the system’s reaction to current input



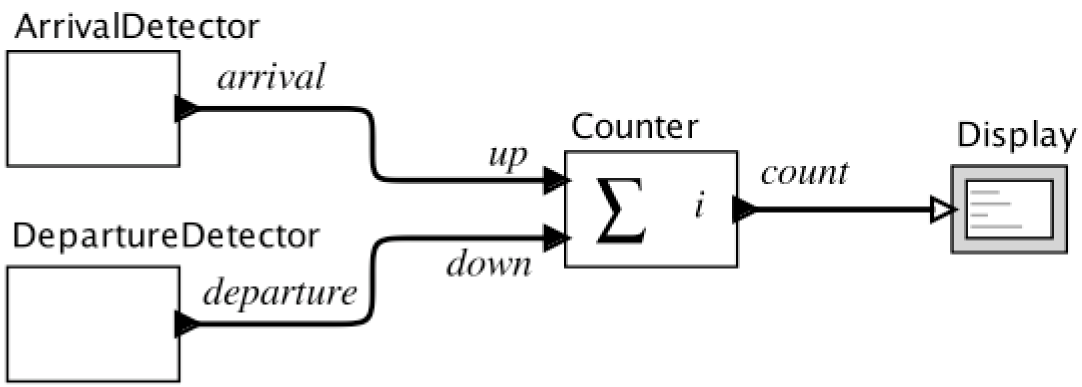
Input and Output

For $t \in \mathbb{R}$ the inputs are in a set

$$Inputs = (\{up, down\} \to \{absent, present\})$$

and the outputs are in a set

$$Outputs = (\{count\} \to \{absent\} \cup \mathbb{N}) \; ,$$



Problem Question

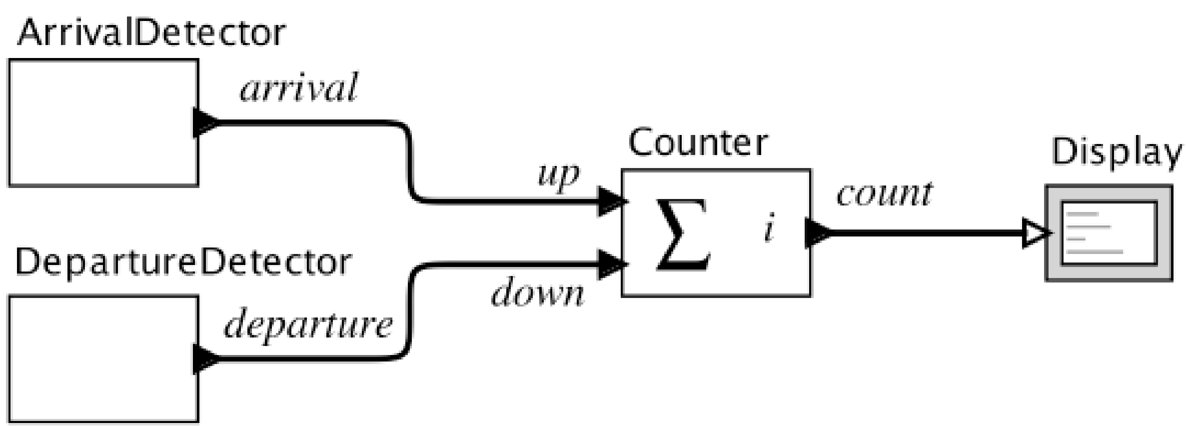
- What are some scenarios that the given parking garage (interface) design does not handle well?

For $t \in \mathbb{R}$ the inputs are in a set

$$Inputs = (\{up, down\} \rightarrow \{absent, present\})$$

and the outputs are in a set

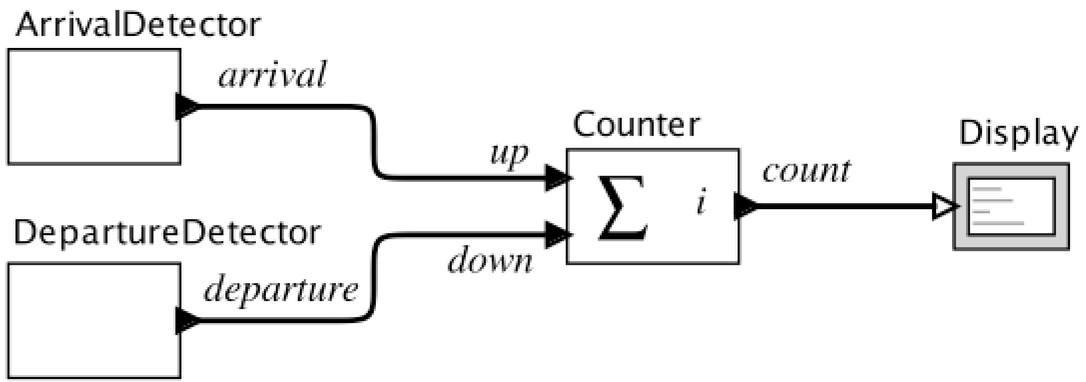
$$Outputs = (\{count\} \rightarrow \{absent\} \cup \mathbb{N}) \ ,$$



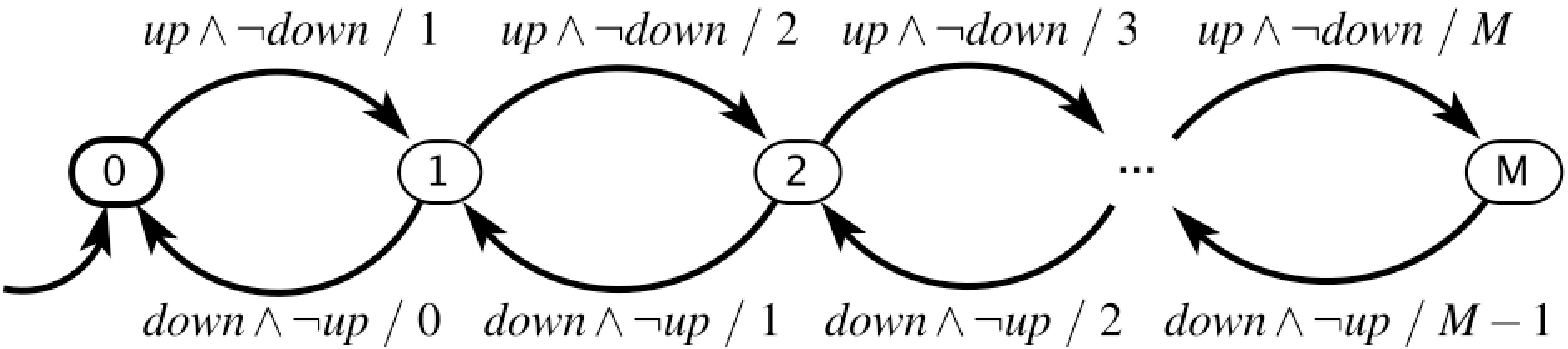
State Space

A practical parking garage has a finite number M of spaces, so the state space for the counter is

$$States = \{0, 1, 2, \cdots, M\} \ .$$



Finite State Machine



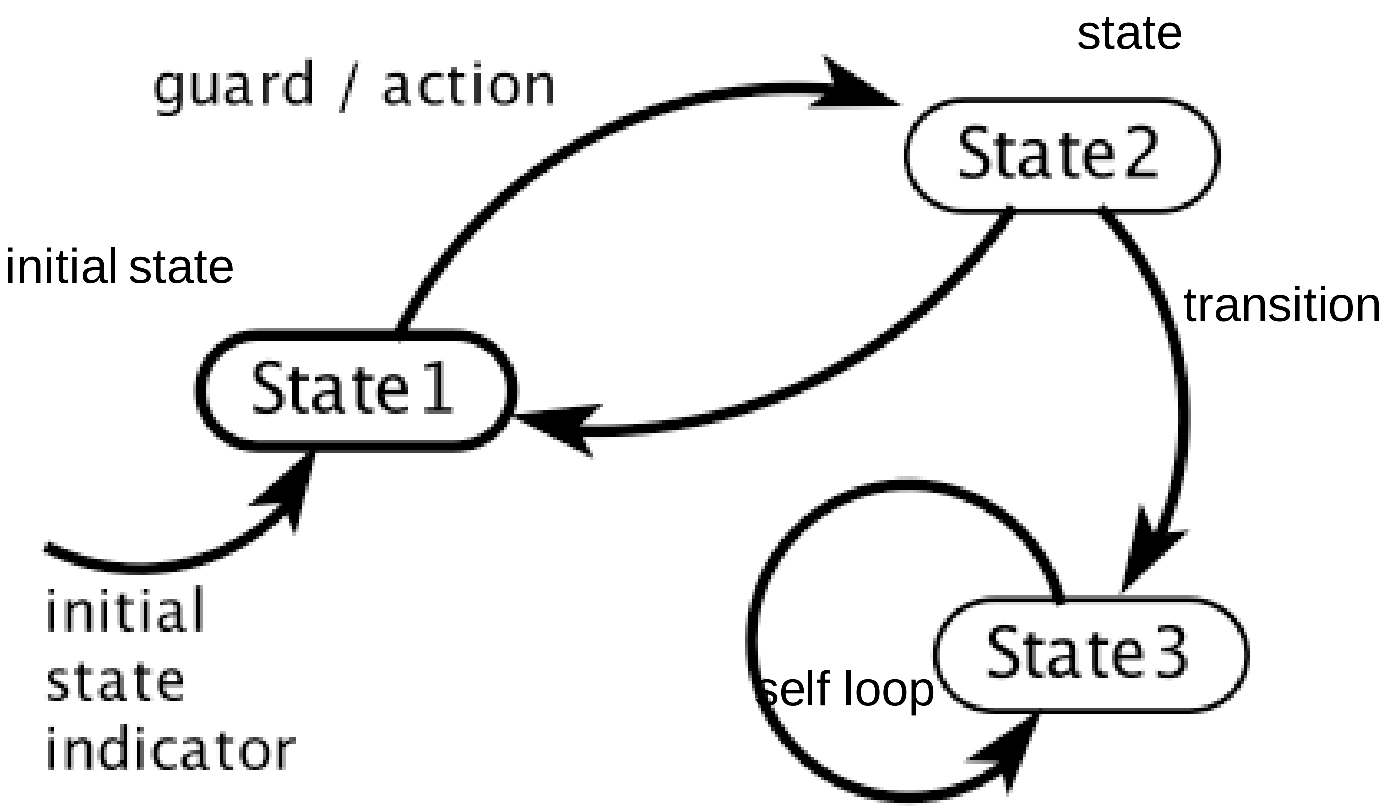
Guard $g \subseteq Inputs$ is specified using the shorthand

$$up \wedge \neg down$$

which means

$$g = \{\{up\}\}\ .$$

FSM Notation

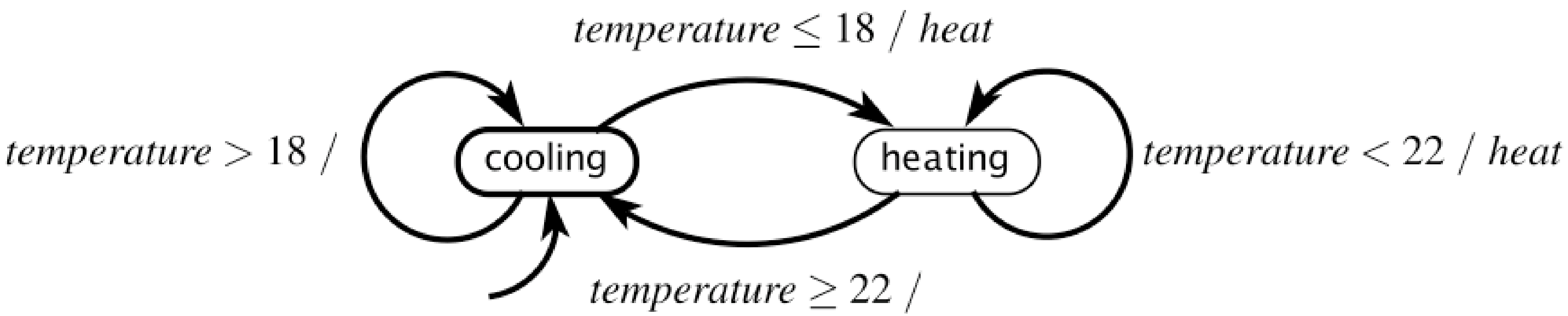


Example of Guard

true	Transition is always enabled.
p_1	Transition is enabled if p_1 is *present*.
$\neg p_1$	Transition is enabled if p_1 is *absent*.
$p_1 \wedge p_2$	Transition is enabled if both p_1 and p_2 are *present*.
$p_1 \vee p_2$	Transition is enabled if either p_1 or p_2 is *present*.
$p_1 \wedge \neg p_2$	Transition is enabled if p_1 is *present* and p_2 is *absent*.

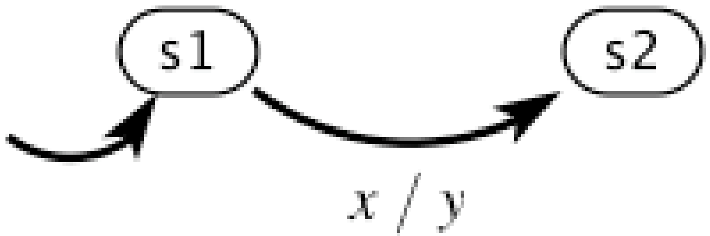
p_3	Transition is enabled if p_3 is *present* (not *absent*).
$p_3 = 1$	Transition is enabled if p_3 is *present* and has value 1.
$p_3 = 1 \wedge p_1$	Transition is enabled if p_3 has value 1 and p_1 is *present*.
$p_3 > 5$	Transition is enabled if p_3 is *present* with value greater than 5.

Example: Thermosta



When does a reaction occur?

input: $x \in \{present, absent\}$
output: $y \in \{present, absent\}$

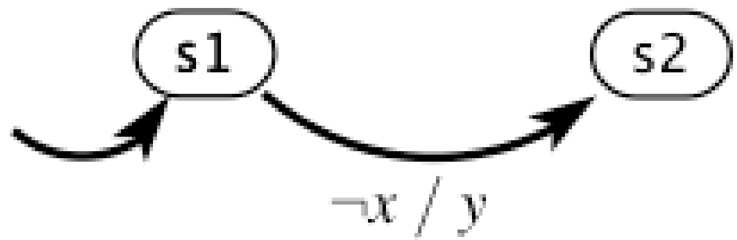


- Suppose all inputs are discrete and a reaction occurs *when any input is present*. Then the above transition will be taken whenever the current state is s1 and *x* is present.

- This is an *event-triggered model*.

When does a reaction occur?

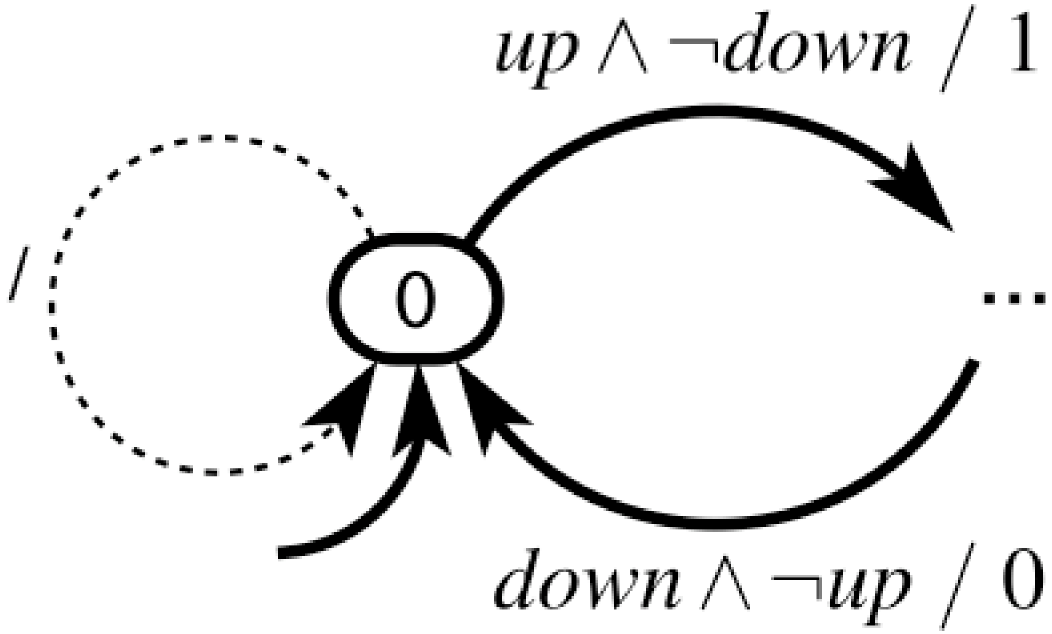
input: $x \in \{present, absent\}$
output: $y \in \{present, absent\}$



- Suppose *x* and *y* are discrete and pure signals. When does the transition occur?

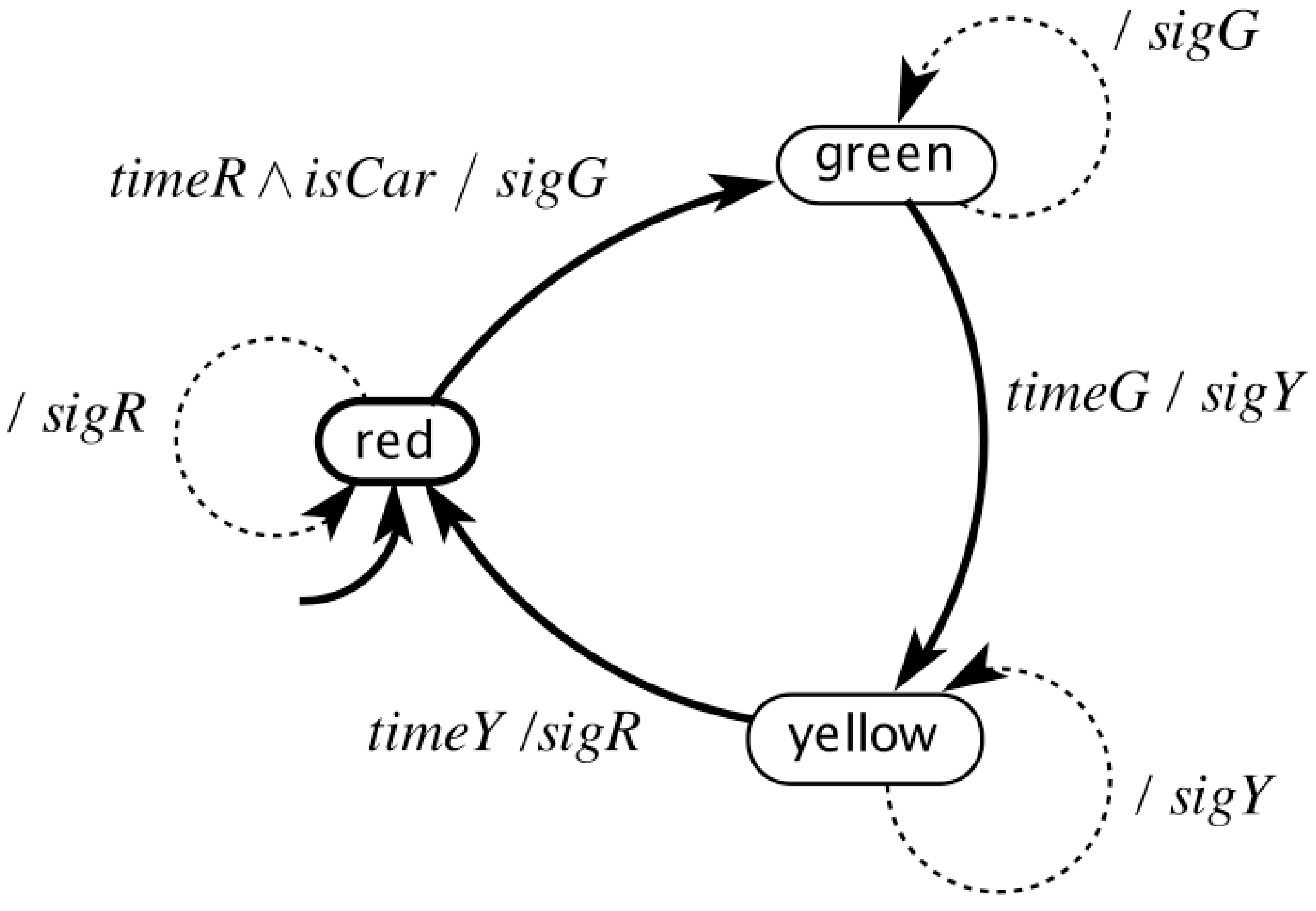
Answer: when the *environment* triggers a reaction and x is absent. If this is a (complete) event-triggered model, then the transition will never be taken because the reaction will only occur when x is present!

Default Transition



- A default transition is enabled if no non-default transition is enabled and it either has no guard or the guard evaluates to true. When is the above default transition enabled?

Example: Traffic Light



More Notation

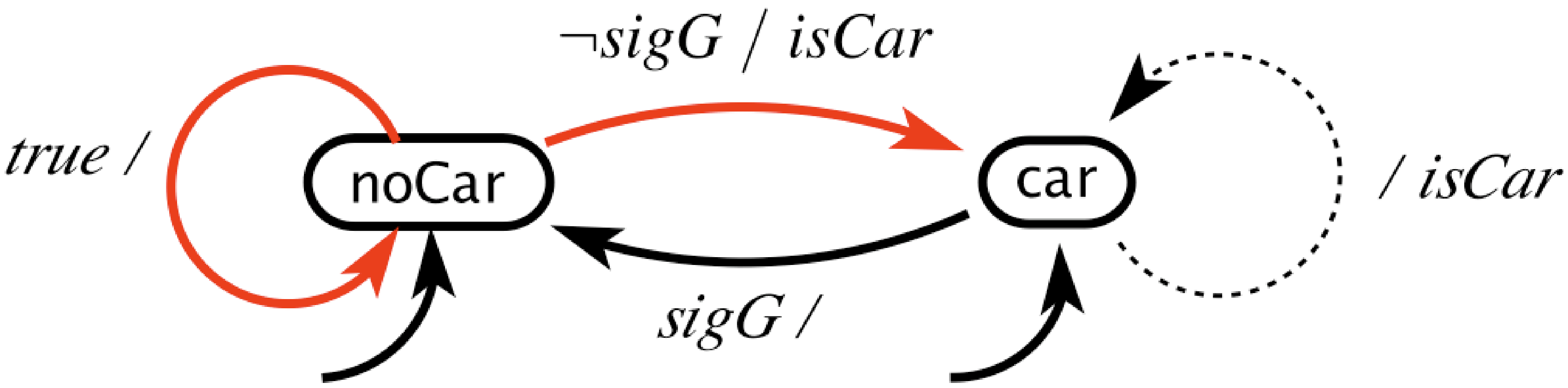
- **Stuttering transition**: (possibly implicit) default transition that is enabled when inputs are absent, that does not change state, and that produces absent outputs.

- **Receptiveness**: For any input values, some transition is enabled. Our structure together with the implicit default transition ensures that our FSMs are receptive.

- **Determinism**: In every state, for all input values, exactly one (possibly implicit) transition is enabled.

Nondeterministic FSM

- Model of the environment for a traffic light, abstracted using nondeterminism:



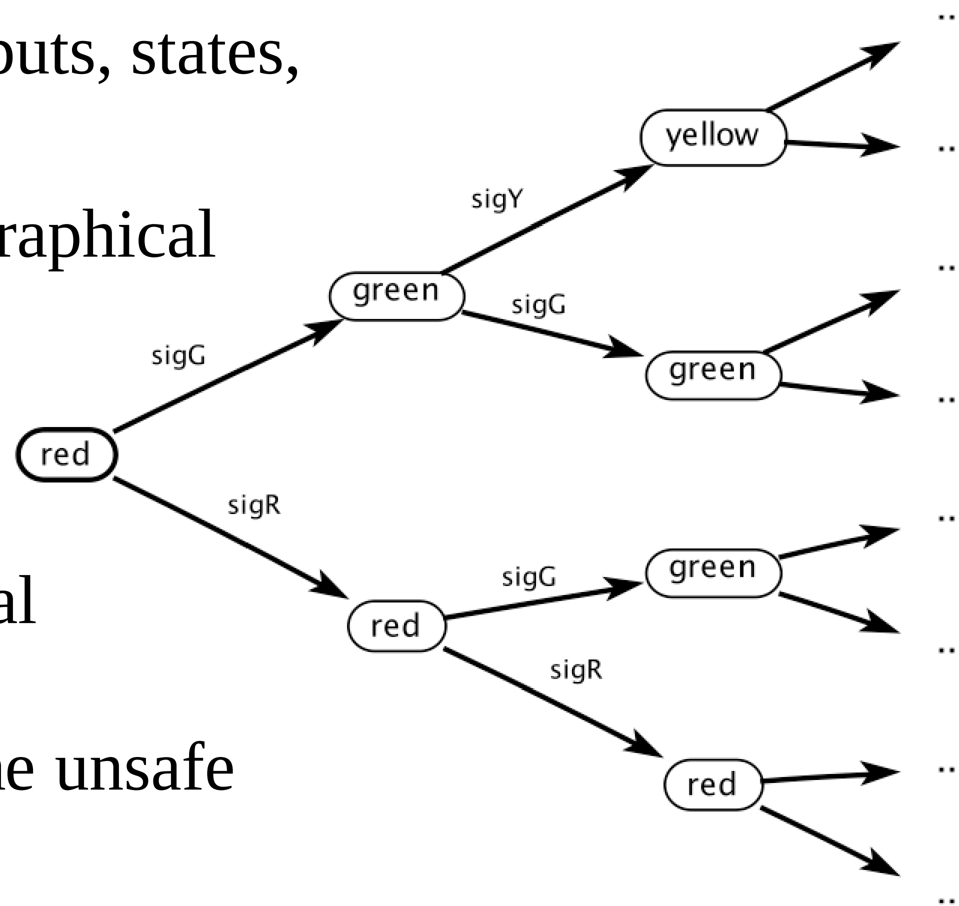
Use of Nonterministic FSM

1. Modeling unknown aspects of the environment or system
   - Such as: how the environment changes a robot's orientation
2. Hiding detail in a *specification* of the system
   - We will see an example of this later (see the text)

- Any other reasons why nondeterministic FSMs might be preferred over deterministic FSMs?

Behaviors and Traces

- FSM **behavior** is a sequence of (non-stuttering) steps.
- A **trace** is the record of inputs, states, and outputs in a behavior.
- A **computation tree** is a graphical representation of all possible traces.

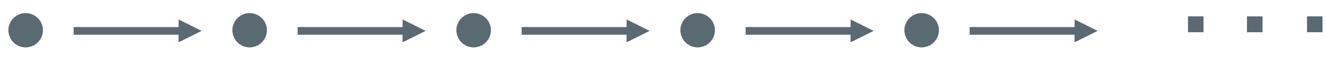


■ FSMs are suitable for formal analysis. For example, **safety** analysis might show that some unsafe state is not reachable.

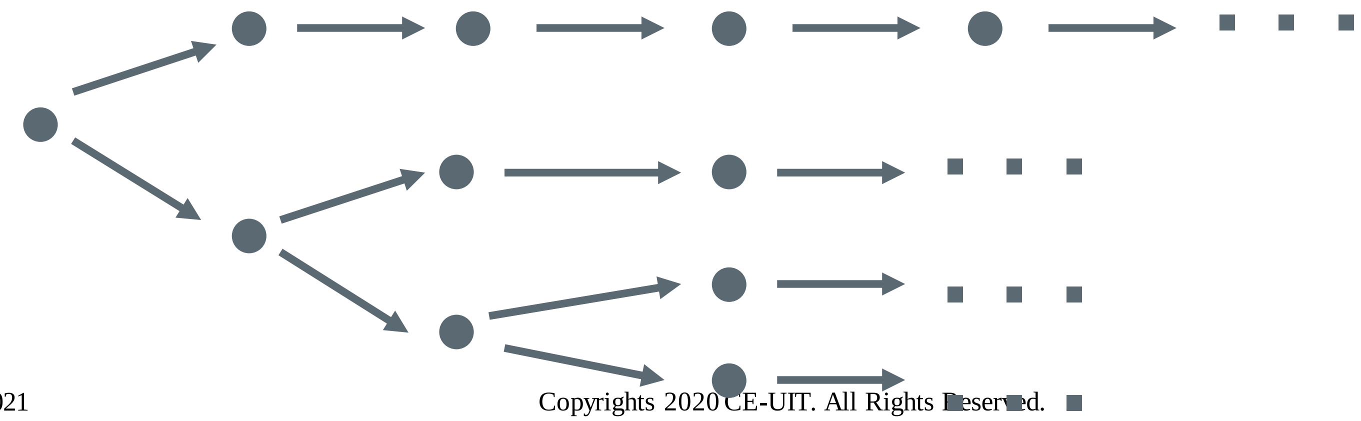
Tree of Computation

- For a fixed input sequence:
  - A deterministic system exhibits a single behavior
  - A non-deterministic system exhibits a **set of behaviors**
    - visualized as a *computation tree*

Deterministic FSM behavior:



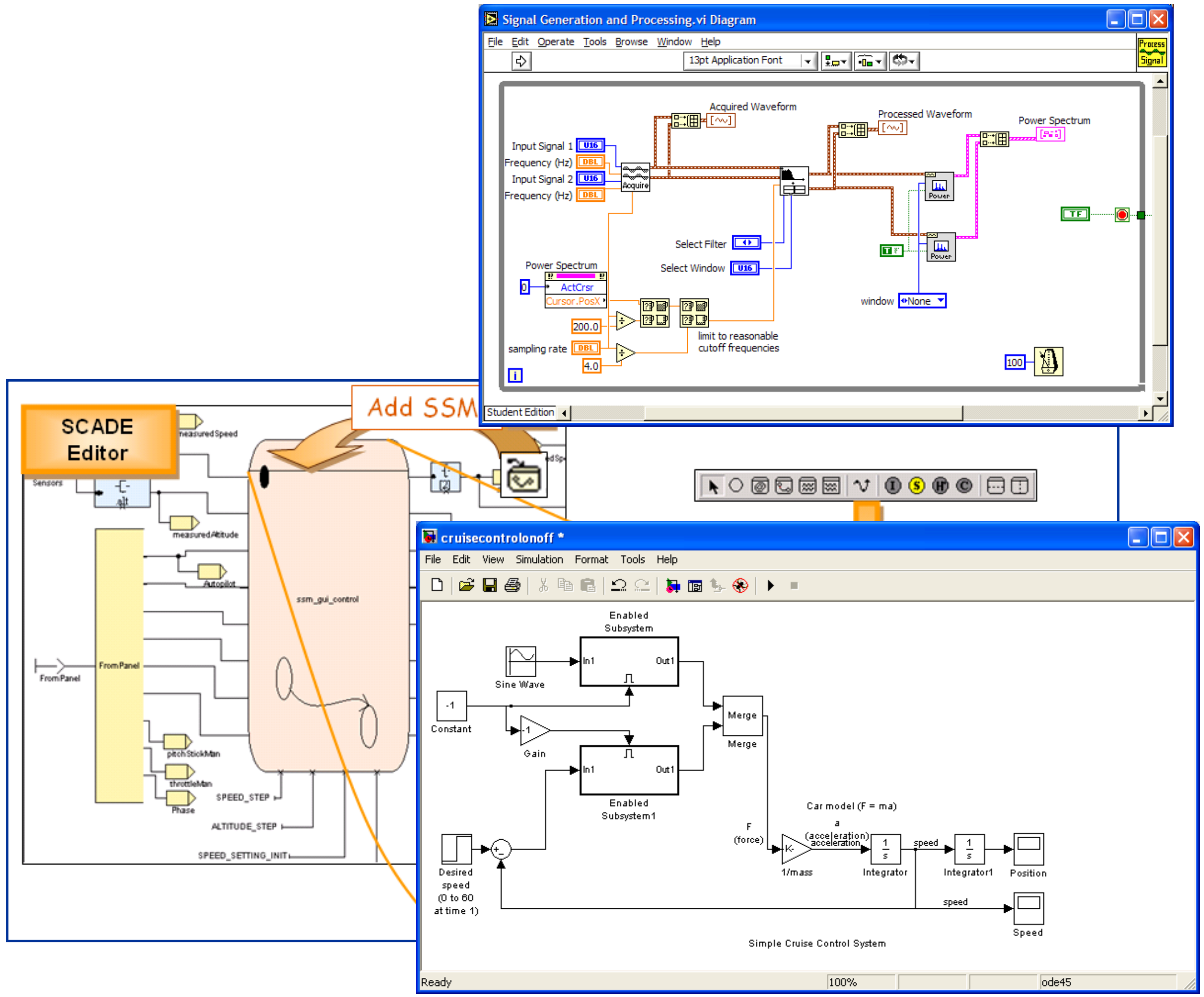
Non-deterministic FSM behavior:



Framework to design Discrete Dynamics

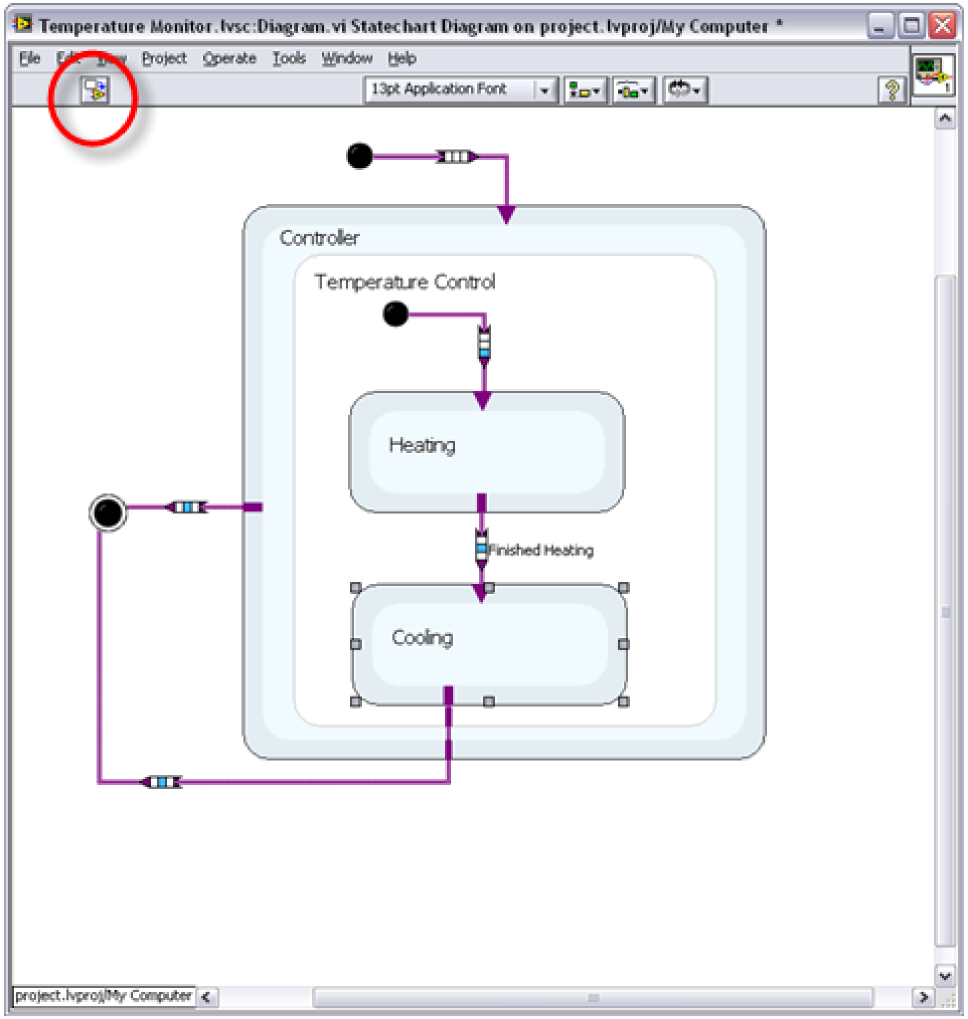
COMPUTER ENGINEERING

- LabVIEW
- Simulink
- Scade
- …

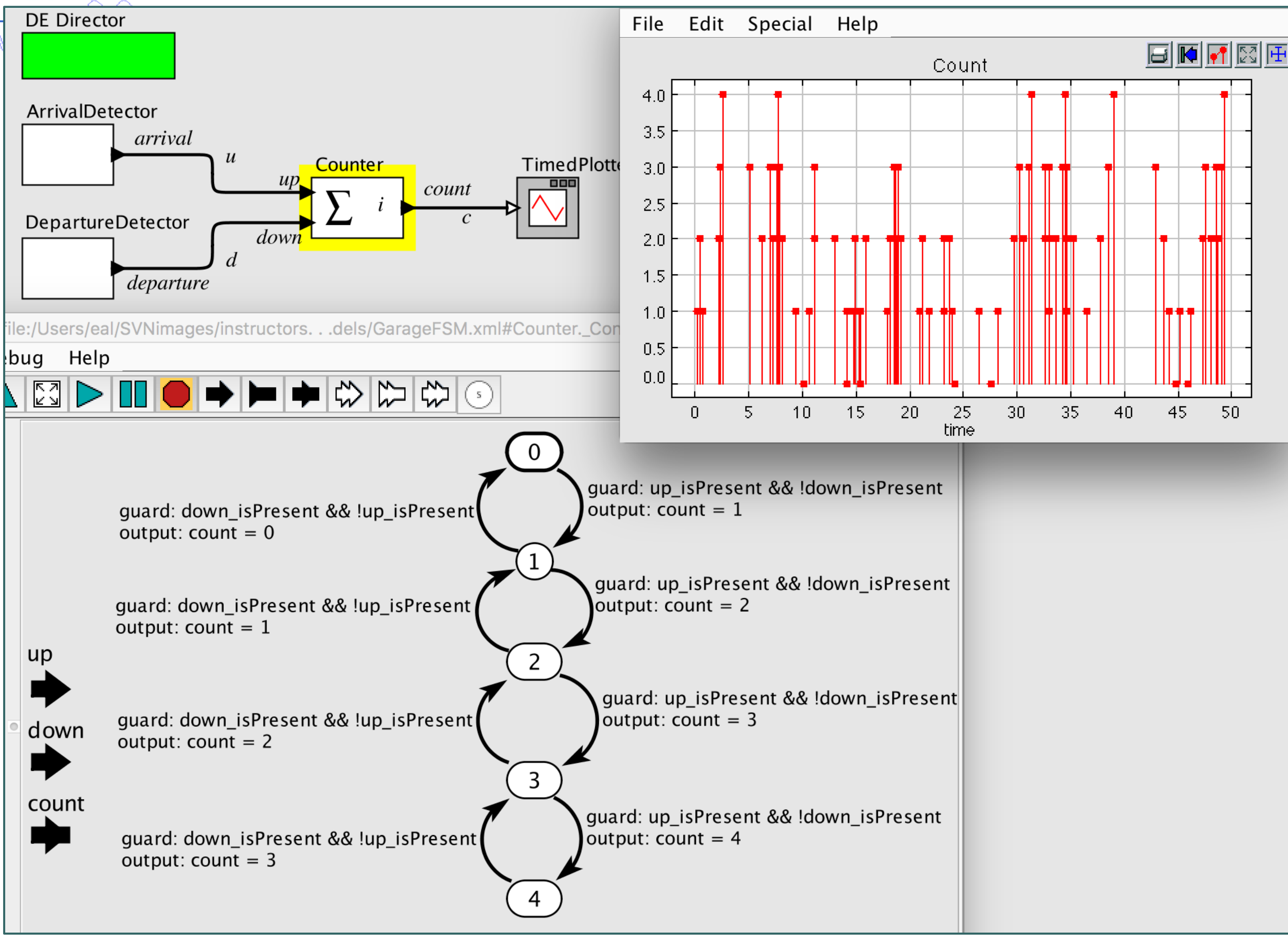


- Reactors
- StreamIT
- Plotemy II model

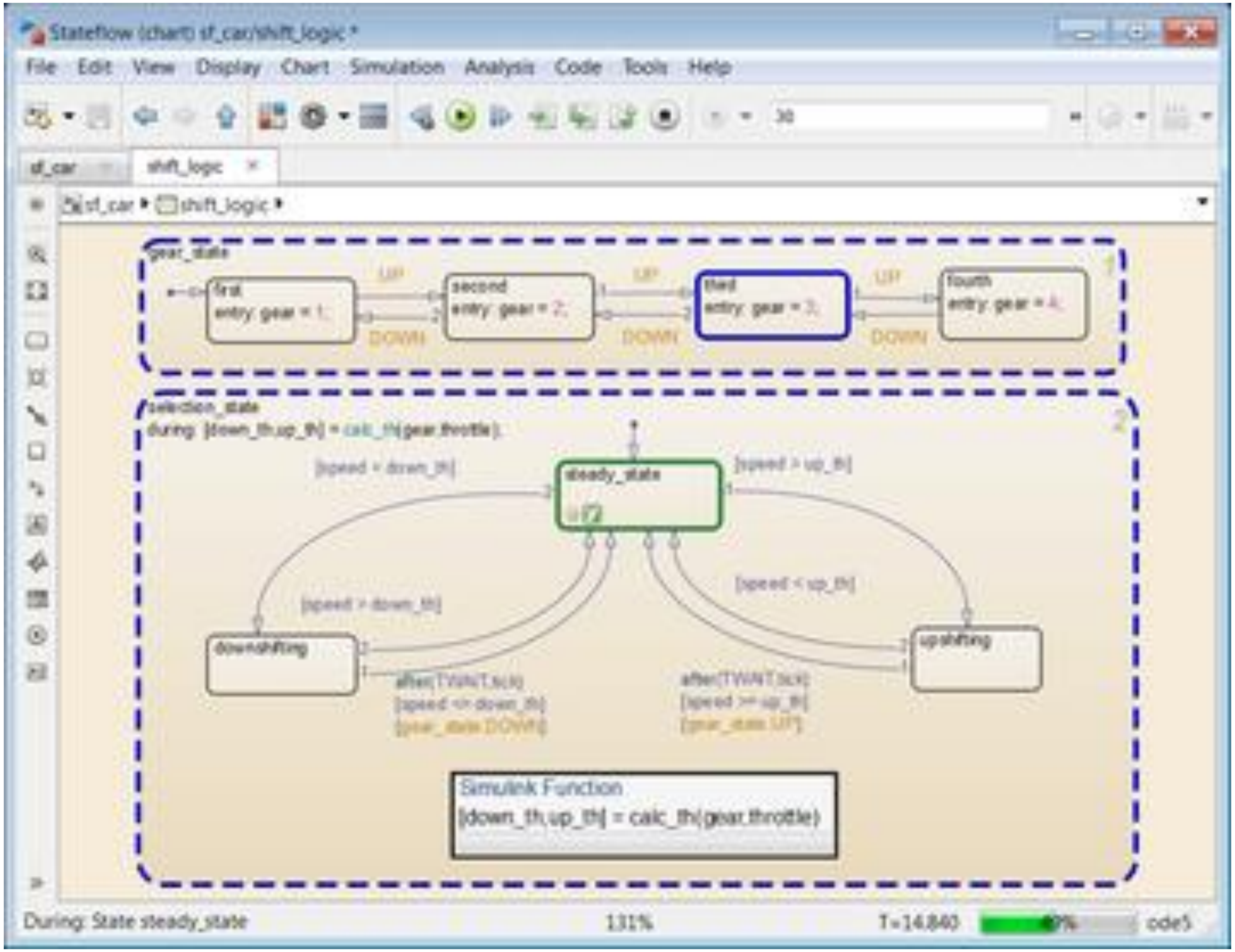
Labview Statechart



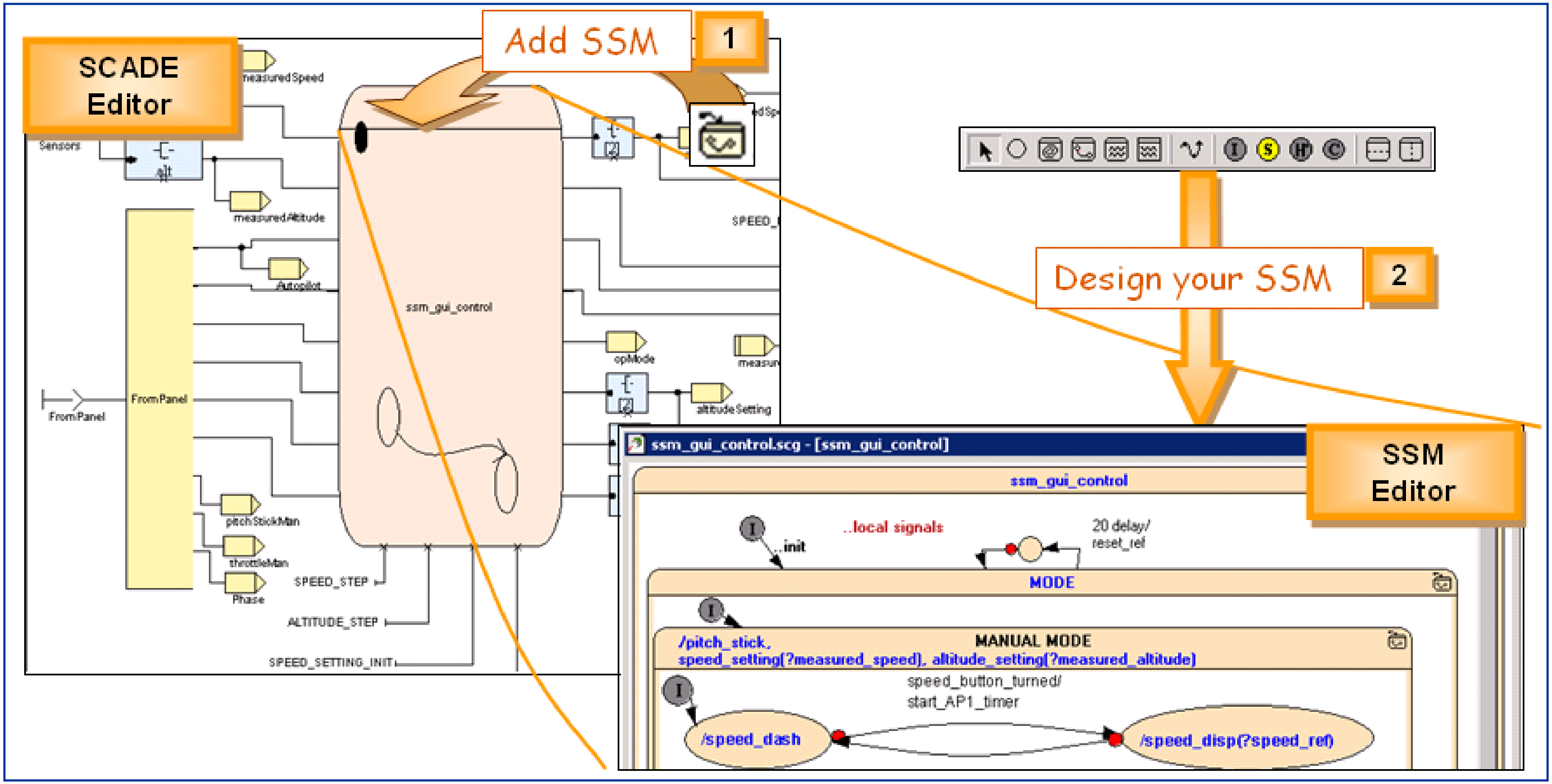
Plotemy II Model



Simulink Stateflow



Scade



SCADE Editor
Add SSM
1
Design your SSM
2
SSM Editor
measuredSpeed
Sensors
alt
measuredAltitude
Autopilot
ssm_gui_control
FromPanel
FromPanel
pitchStickMan
throttleMan
Phase
SPEED_STEP
ALTITUDE_STEP
SPEED_SETTING_INIT
opMode
altitudeSetting
ssm_gui_control.scg - [ssm_gui_control]
ssm_gui_control
..local signals
init
20 delay/
reset_ref
MODE
MANUAL MODE
/pitch_stick,
speed_setting(?measured_speed), altitude_setting(?measured_altitude)
speed_button_turned/
start_AP1_timer
/speed_dash
/speed_disp(?speed_ref)

Q&A