Introduction to Engineering Using Robotics Experiments

Finite State Machine

References
http://en.wikipedia.org/wiki/Finite-state_machine

Lecture 06

Yinong Chen

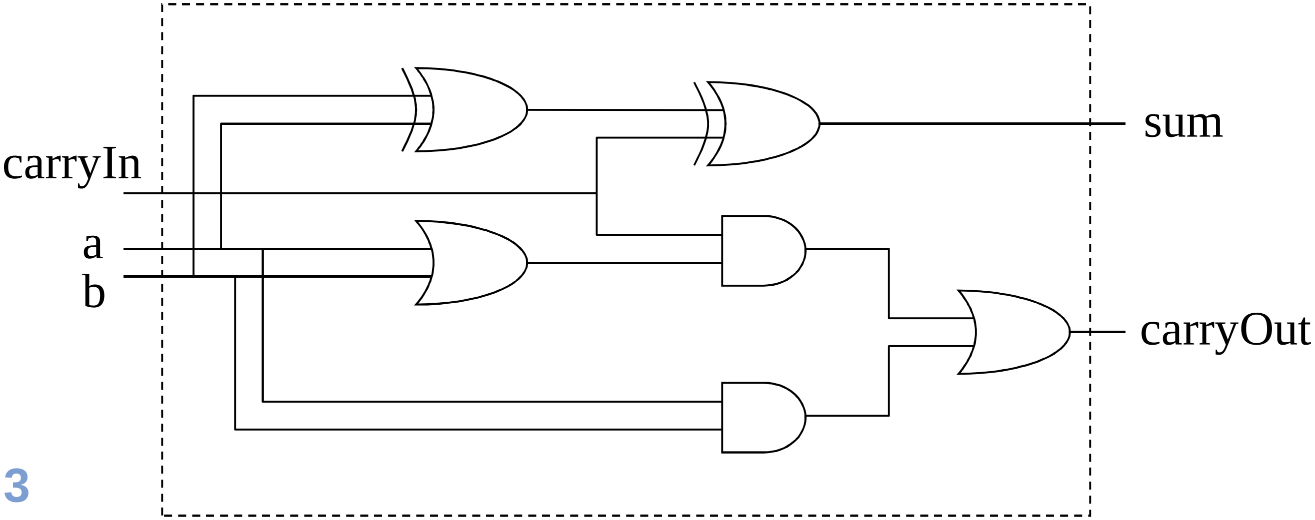
Roadmap: Evaluation in Design Process

1. Combinational and Sequential Circuits
2. Stateless Vending Machine Design
3. Finite State Machine
4. Examples of Finite State Machines

Combinational Circuits

input			output	
a	b	CarryIn	CarryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

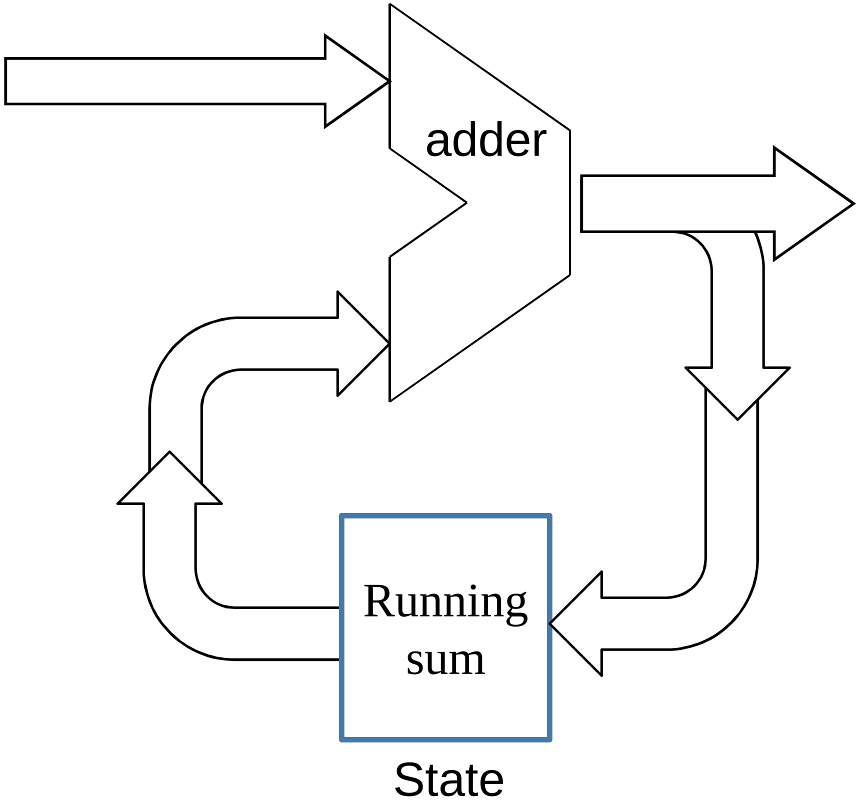
- Stateless: information cannot be stored in the circuit;
- Output is determined by input only;
- Truth table fully specifies the function



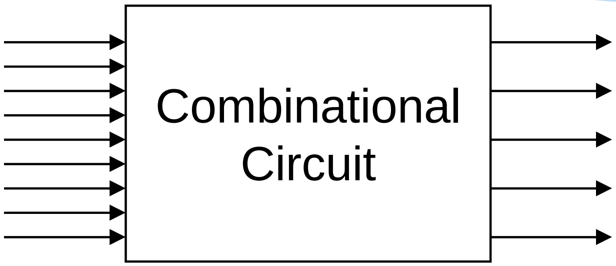
Sequential Circuits

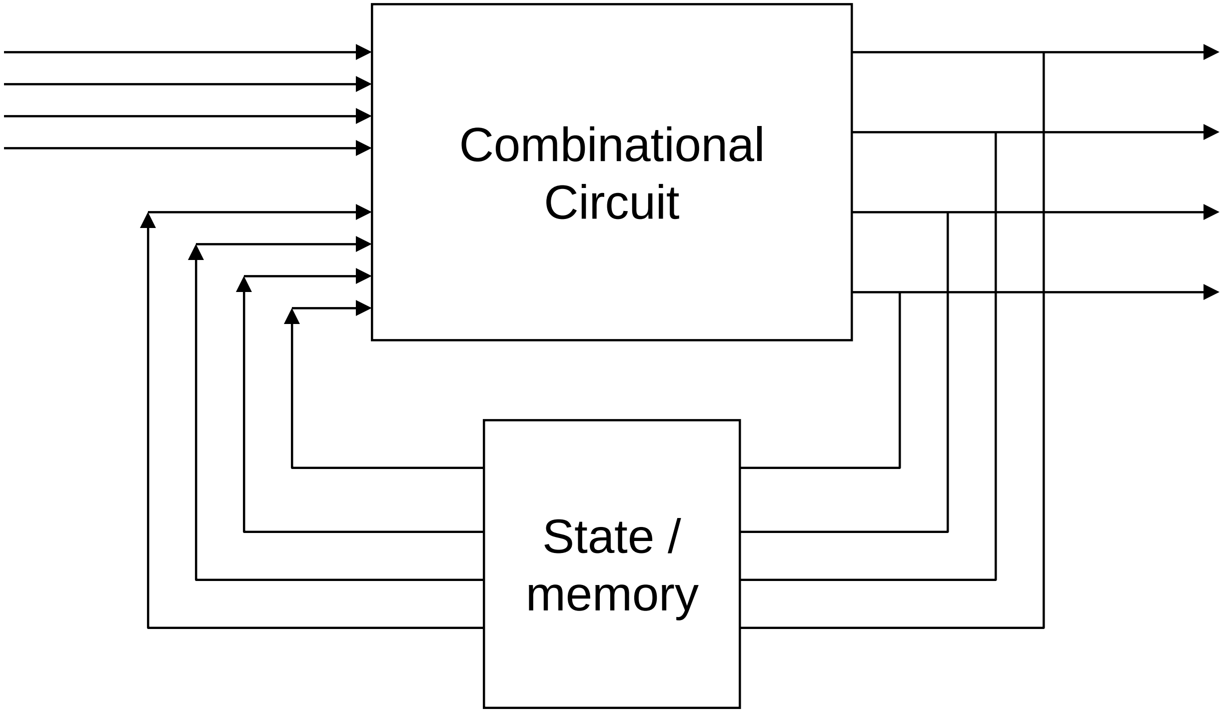
- The circuit stores state (internal values calculated in the past)
- Output is determined by input and state;
- Finite state machine specifies the function.

Example:
an accumulator



Combinational and Sequential Circuits





Finite State Machine (FSM)

- Truth table serves as the specification of
  - Combinational circuit (hardware)
- An Finite State Machine serves as the specification of
  - a sequential circuit (hardware), to be taught in CSE 120, and
  - **an event-driven program (software)**

Model For a Stateless Vending Machine

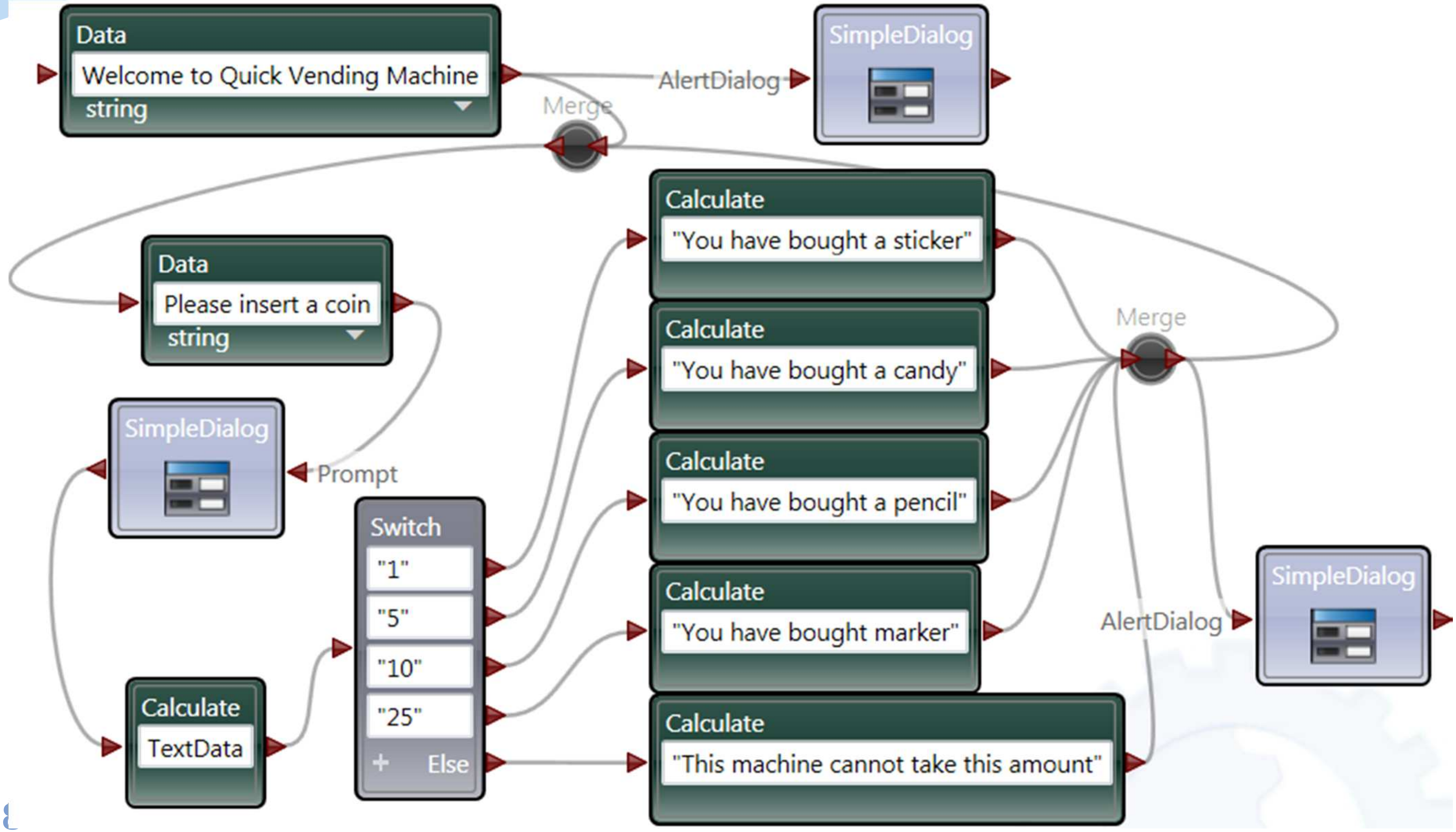
- For a given input, it gives an output immediately

- Problem Definition: Use these US currency coins to purchase products in the machine;
- Parameters: coins and products
- Range of values for each parameter:
  - Coins: 1, 5, 10, 25
  - Products : sicker, candy, pencil, and marker
- Constraints/Relationships /Solution (function table):

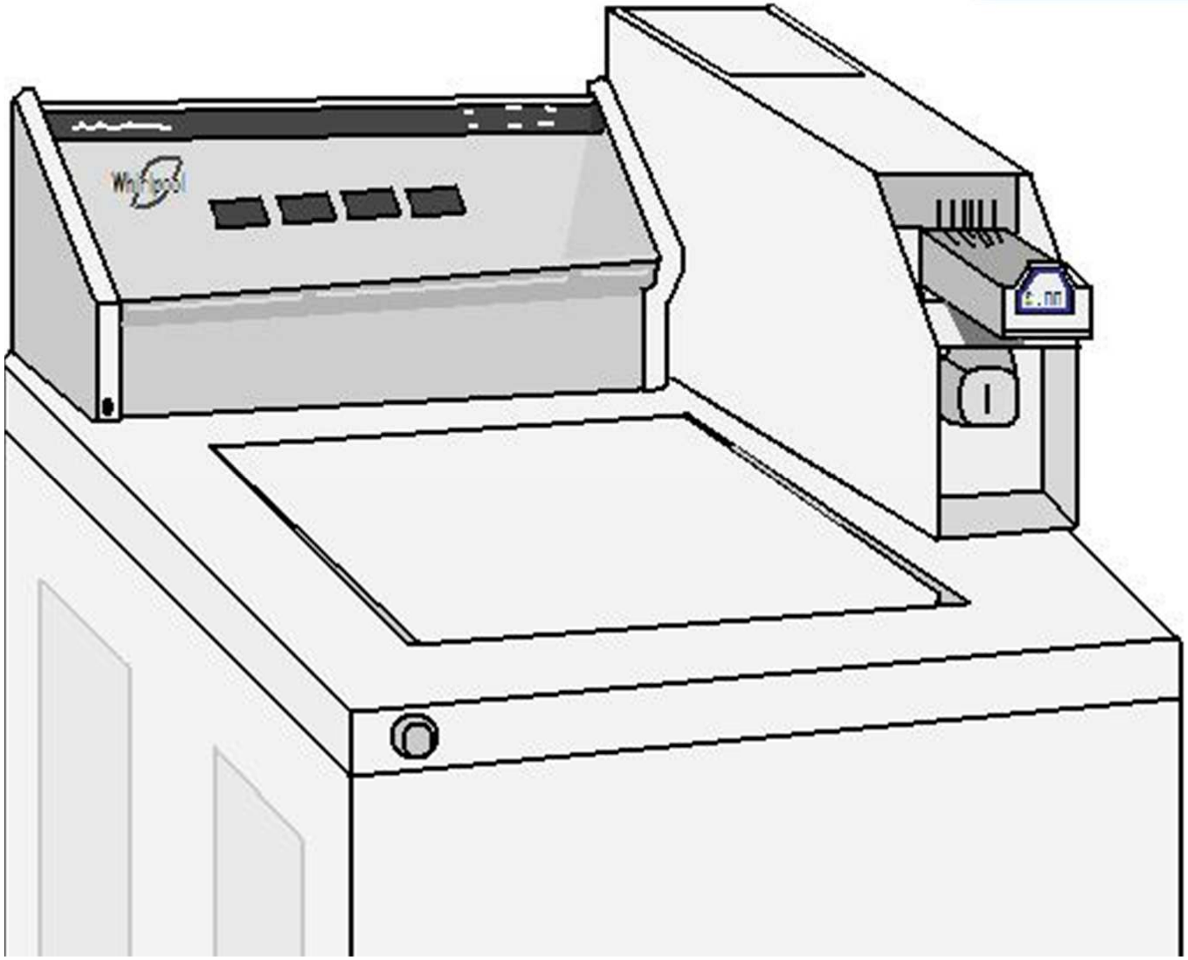
Coins	Penny (1)	Nickel (5)	Dime (10)	Quarter
Products	Sicker	candy	pencil	marker

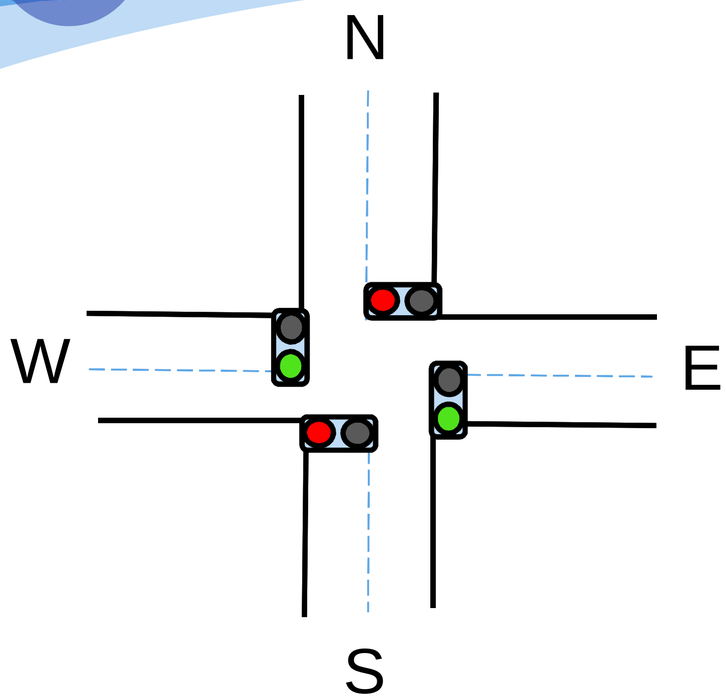
A Stateless Vending Machine

- VPL Implementation: No variable is used



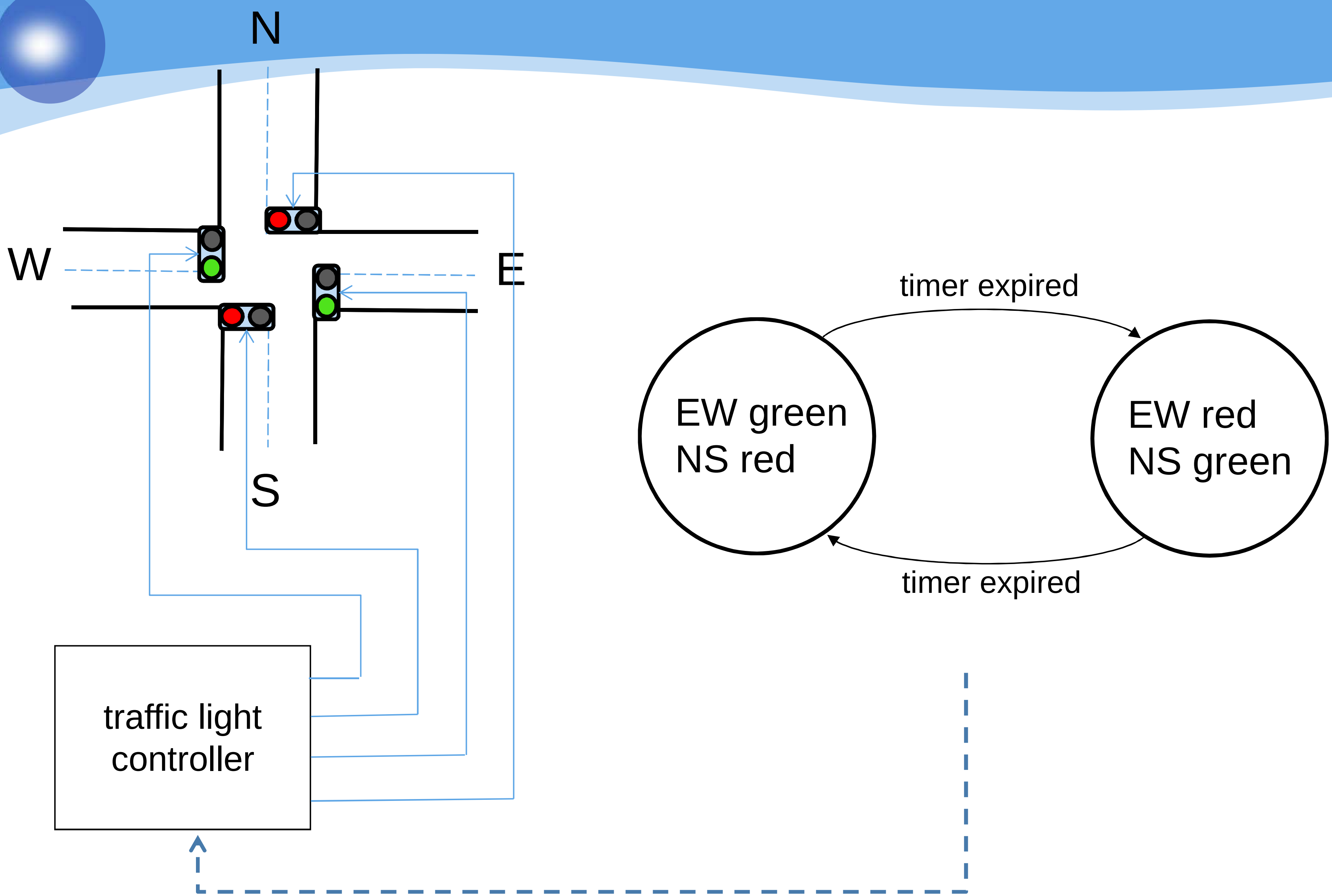
Why does a coin-operated washing machine take all coins at the same time?





A four-way intersection has red/green traffic lights that are controlled with timers.

Traffic can only move in one direction at a time: NS)North-South) or EW (East-West).

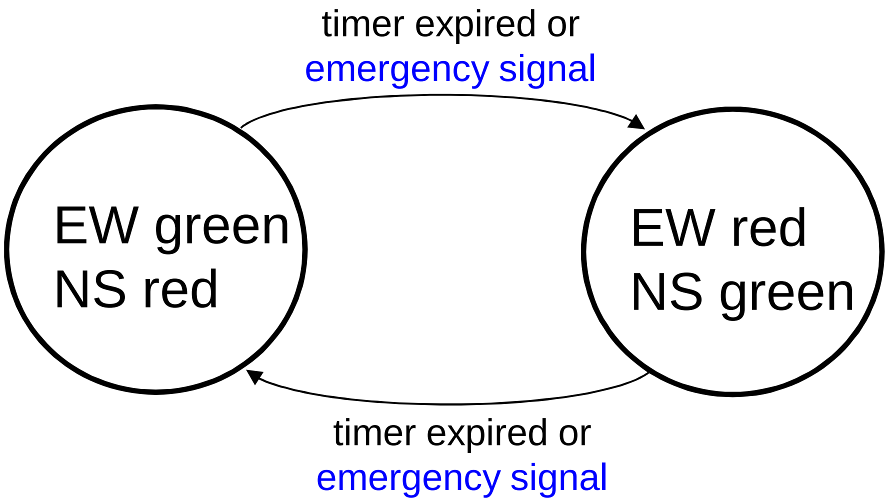

N
W
E
S
traffic light controller
timer expired
EW green NS red
EW red NS green
timer expired

A **Finite-State Machine** (FSM)

is a model of the discrete dynamics of a system that has a finite number of discrete **states**. **Transitions** between states are caused by **events**, such as:

- the expiration of a timer
- a change in a sensor value

state diagram



state table

current state	event	next state
EW grn/NS rd	timer exp	EW rd/NS grn
EW rd/NS grn	timer exp	EW grn/NS rd

The Traffic Lights by Canary Wharf Tower, East London

Finite State Machine (FSM)

- A Finite State Machine is a mathematical model consisting of a finite number of states, transitions between states, inputs, and outputs.
- Finite State Machines are designed to respond to a **sequence** of inputs (events), such as
  - coin insertions into a vending machine
  - mouse-clicks/key strikes during a program's execution
  - The arrival of individual characters from a string
- Each input causes a transition from one to another state
- An output can be associated to an input

Finite-State Machines are often used to design control systems...

Open Close

No states required

When button pressed:
If state==open
then close
else open

Required states

Finite-State Machines are often used to design control systems...

A garage door opening system

If the door is closed and I press the button (touch sensor), the door begins to move up.

When it reaches the top, the door activates a limit switch (a touch sensor) and stops.

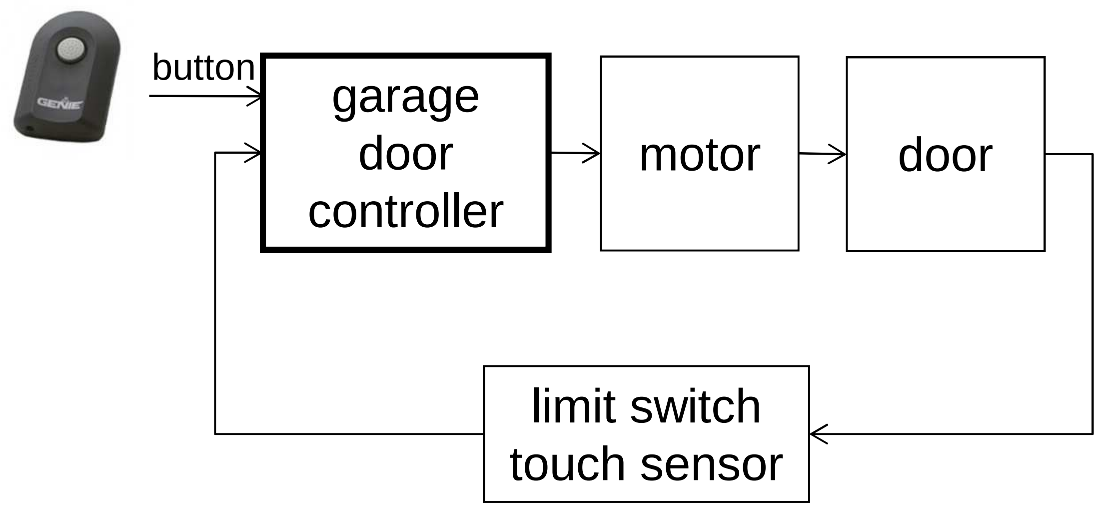
If the door is open and I press the button, the door begins to move down.

When it reaches the bottom, the door activates another limit switch and stops.

Finite-State Machines are often used to design control systems...

A garage door opening system

block diagram



...we want to design the controller...

Finite-State Machines are often used to design control systems...

A garage door opening system

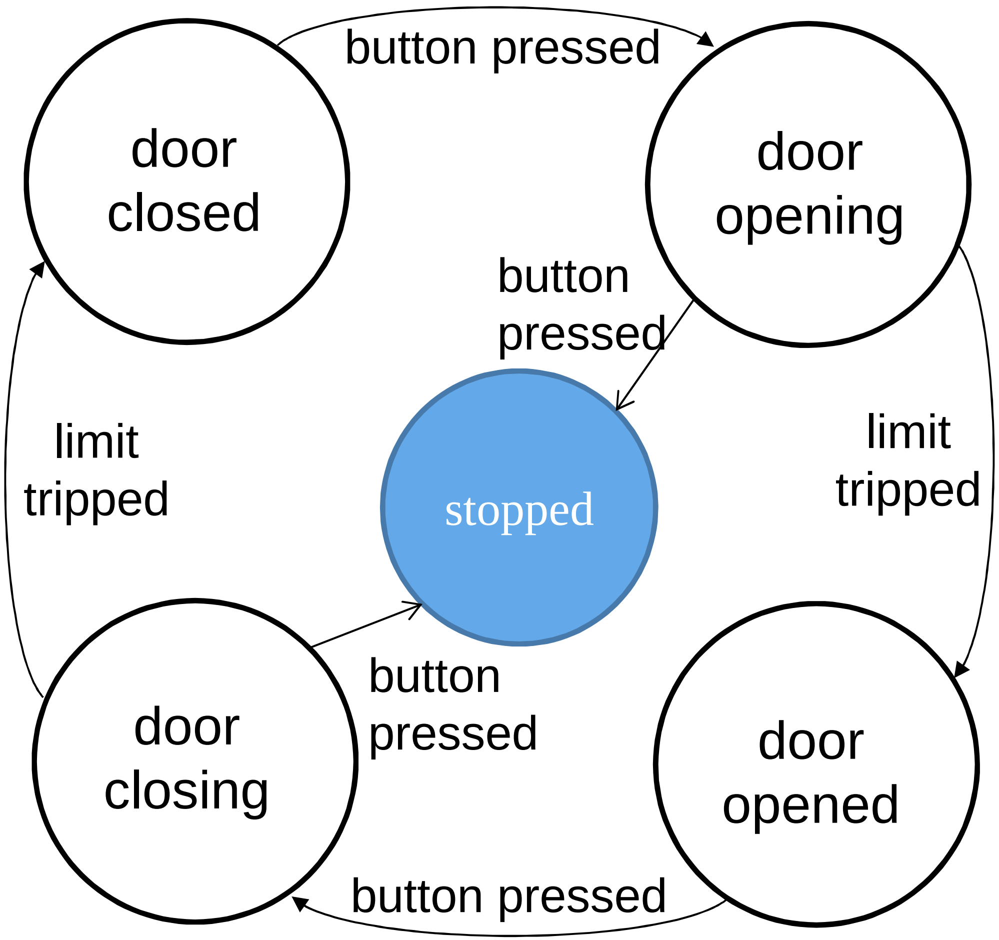
states

- door closed
- door open
- door closing
- door opening

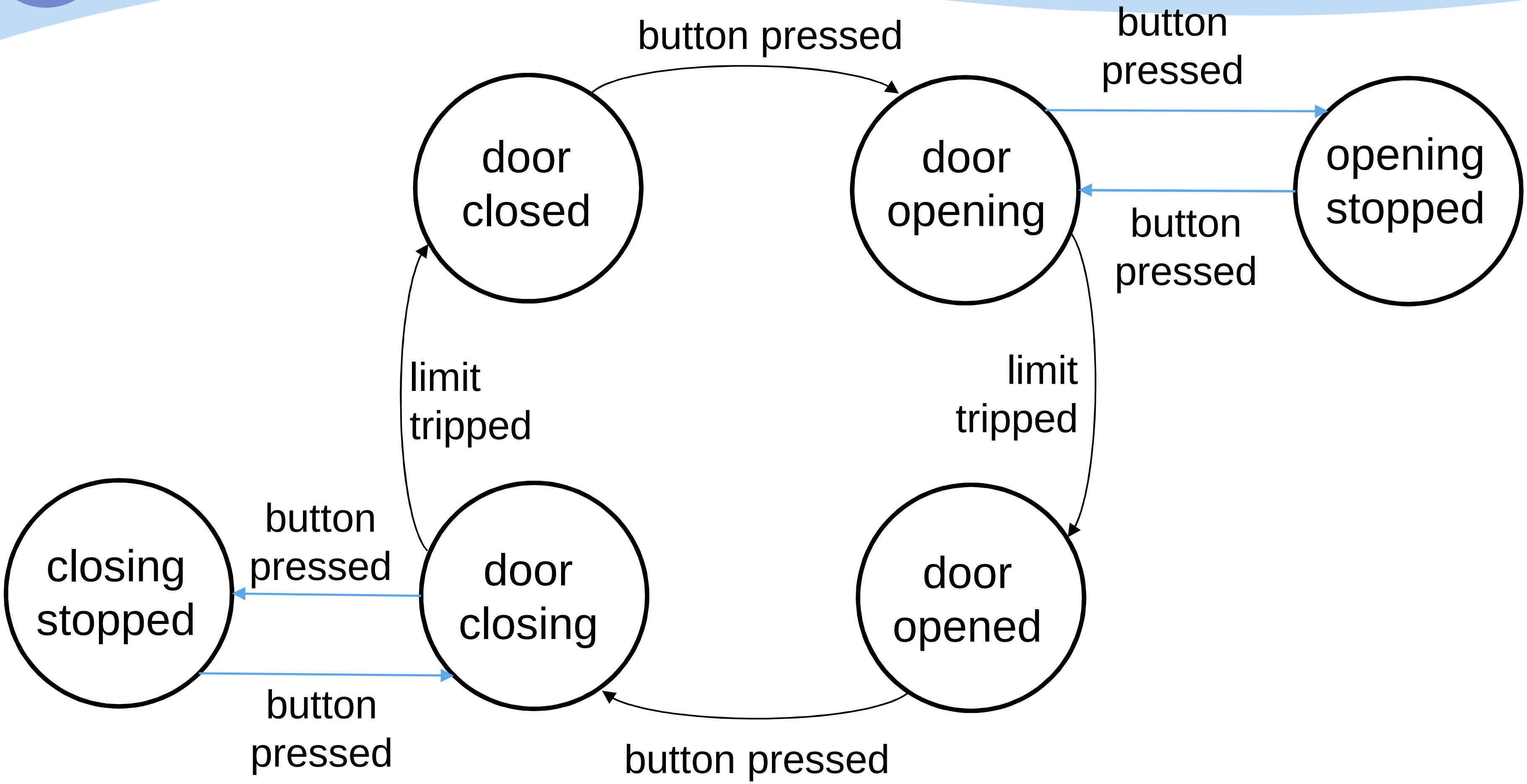
events

- button press
- limit switch touched
  (closing finished or opening finished)

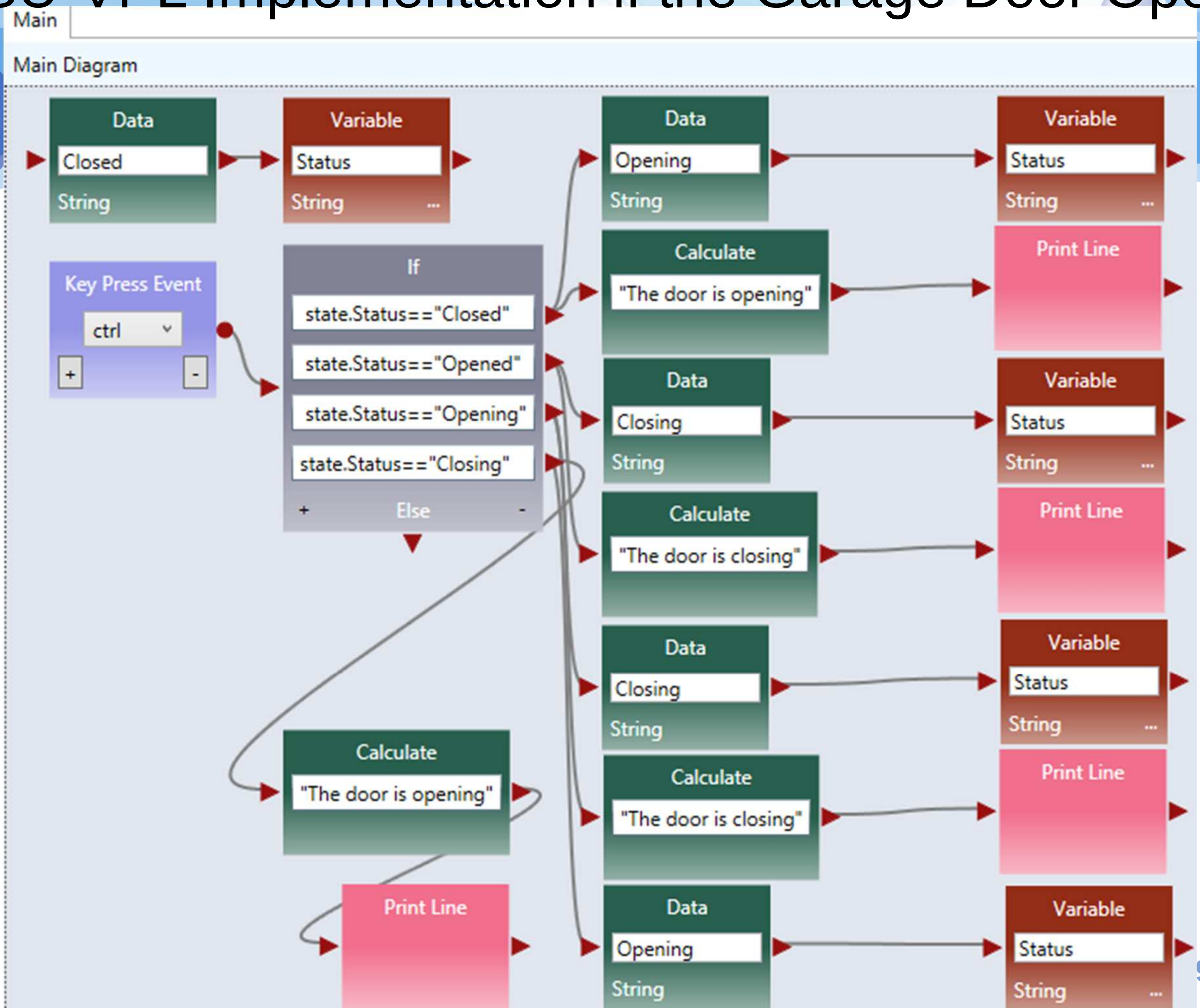
Finite-State Machines are often used to design control systems...



Finite-State Machines are often used to design control systems...

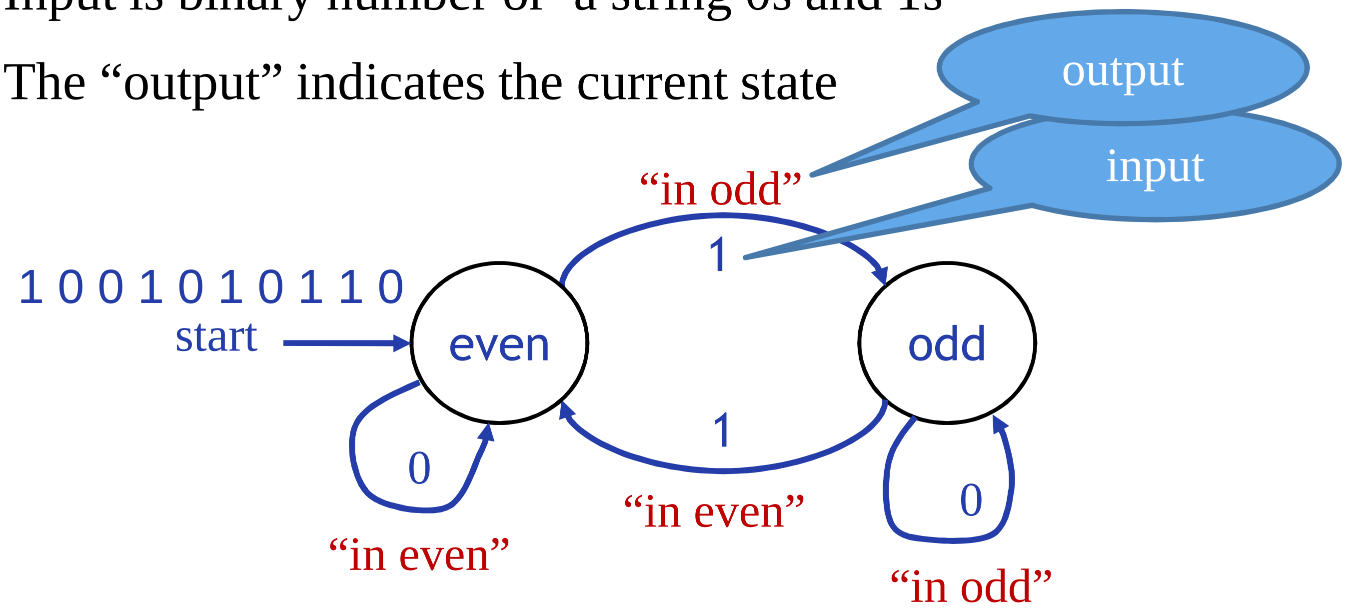


ASU-VPL Implementation if the Garage Door Opener



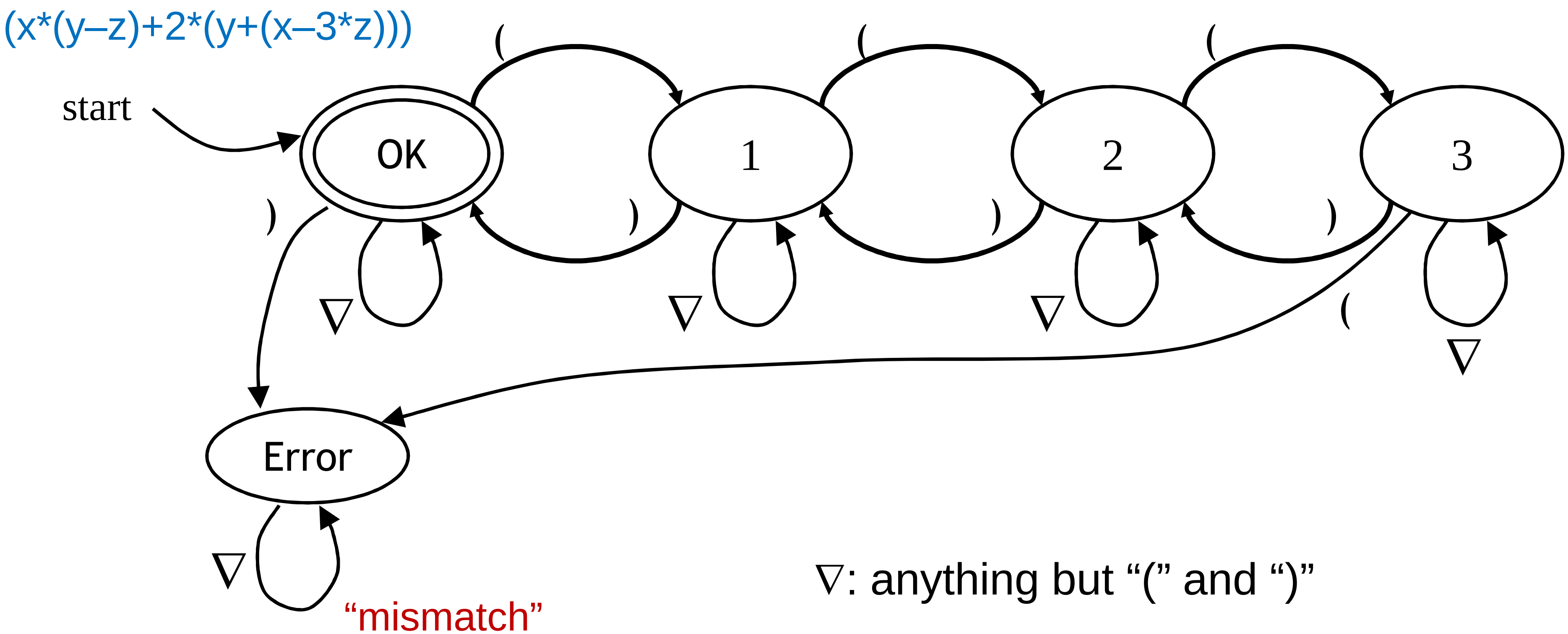
Example 1: Detecting Even or Odd

- The following FSM determines whether the **number of 1s is even or odd**, for a given binary number, e.g., 1001010110
  - Circles represent states; arrows represent transitions
  - Input is binary number or a string 0s and 1s
  - The "output" indicates the current state

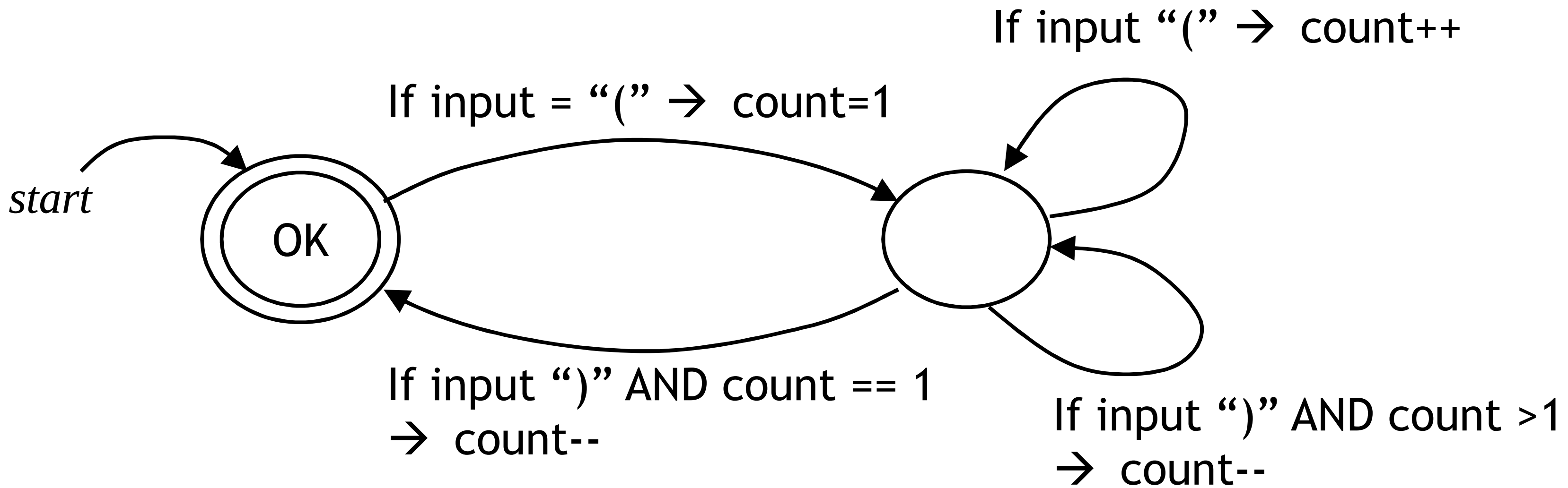


Example 2: Nested Parenthesis

- The following example tests whether parentheses are properly nested (up to 3 deep)



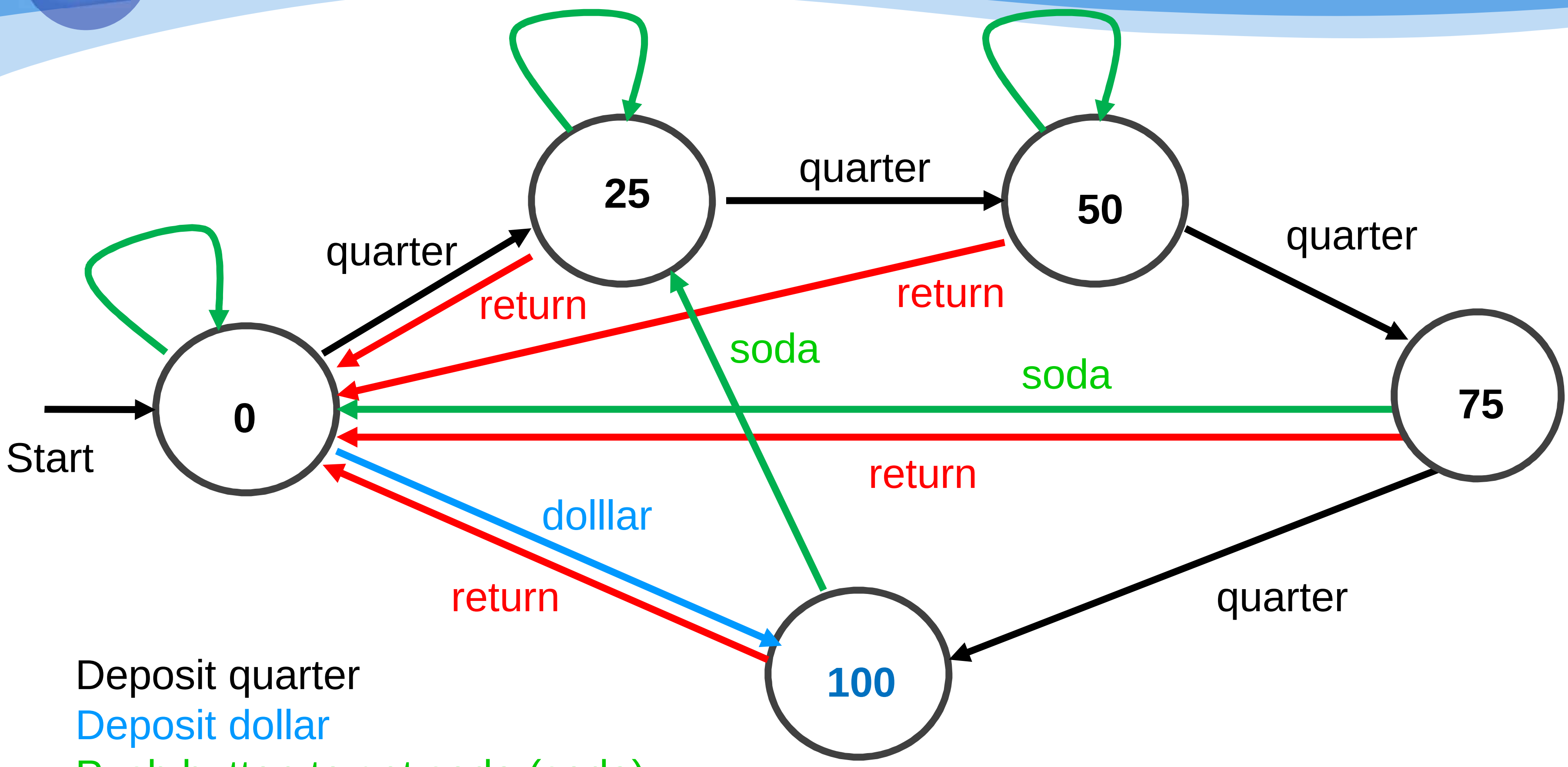
Nested Parentheses Using an Additional Variable



Example 3: FSM Vending Machine

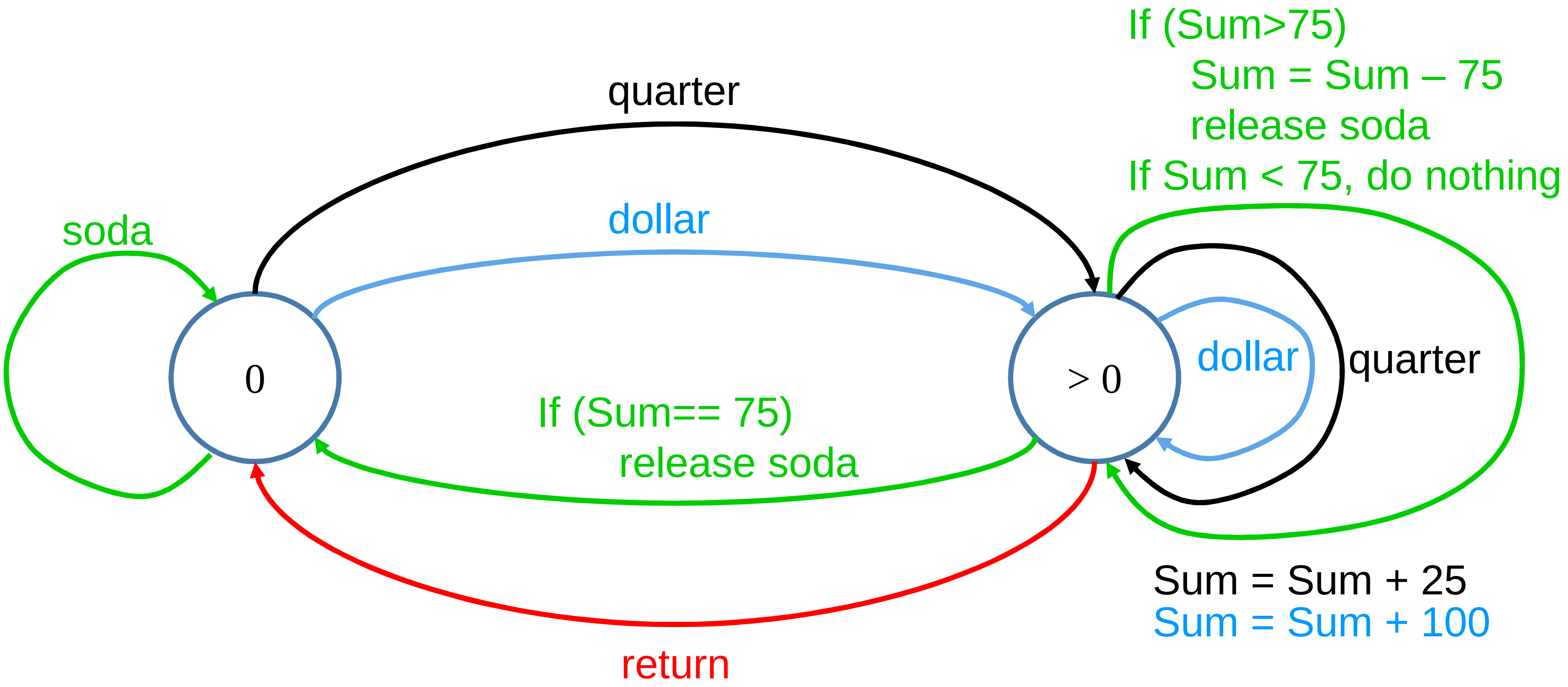
- Takes quarters and dollars only
- Maximum deposit is $1 (or four quarters)
- Sodas cost $0.75
- Possible Inputs (Events):
  - Deposit quarter (25)
  - Deposit dollar (100)
  - Push button to get soda (soda)
  - Push button to get money returned (ret)
- States: 0, 25, 50, 75, 100, and state transits on input

Example 3: FSM Vending Machine

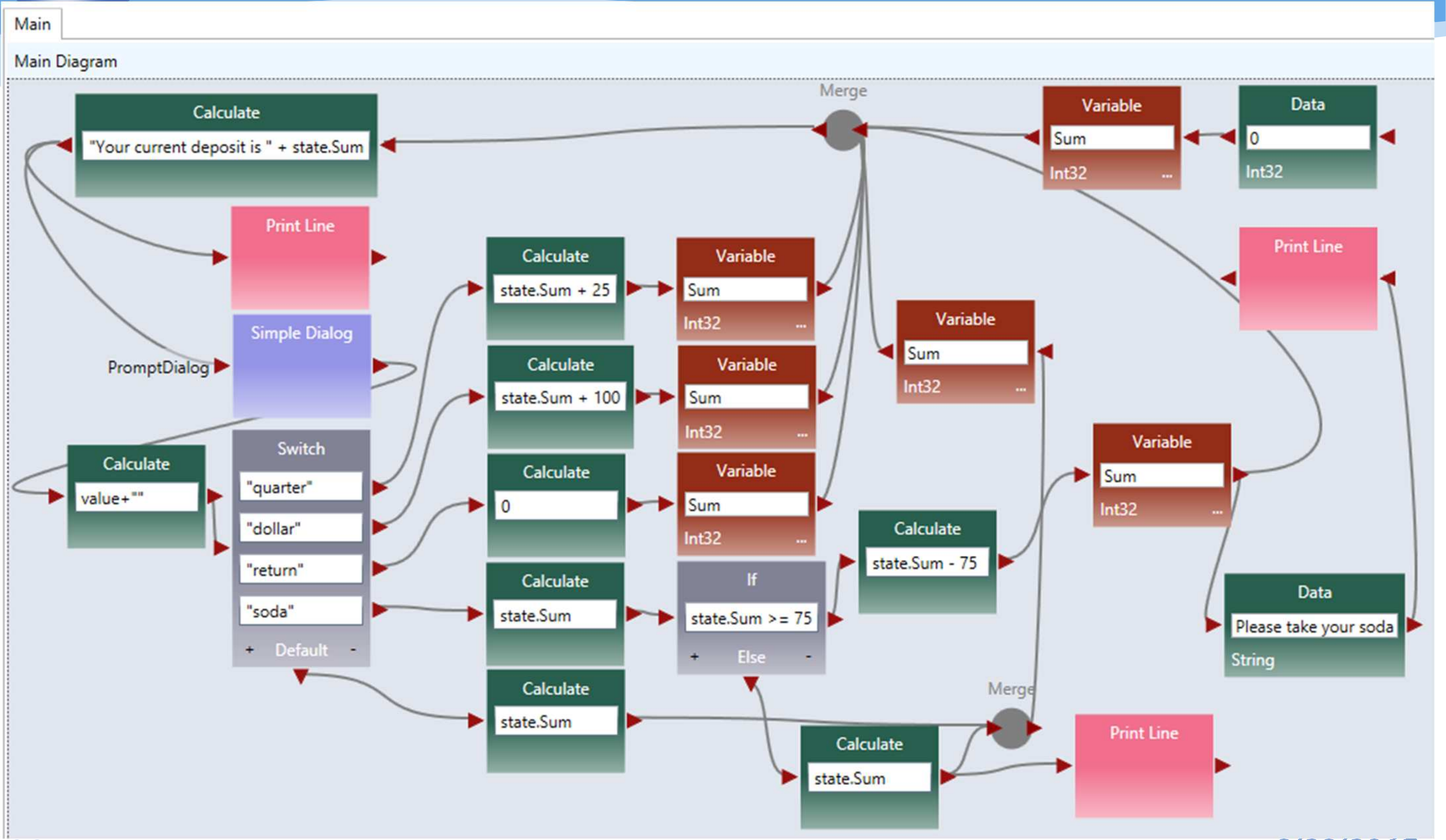


Deposit quarter
Deposit dollar
Push button to get soda (soda)
Push button to return money (ret)

FSM With Additional Memory



Example 3: FSM Vending Machine



The Project...

An **autonomous mobile robot** must navigate through a maze.

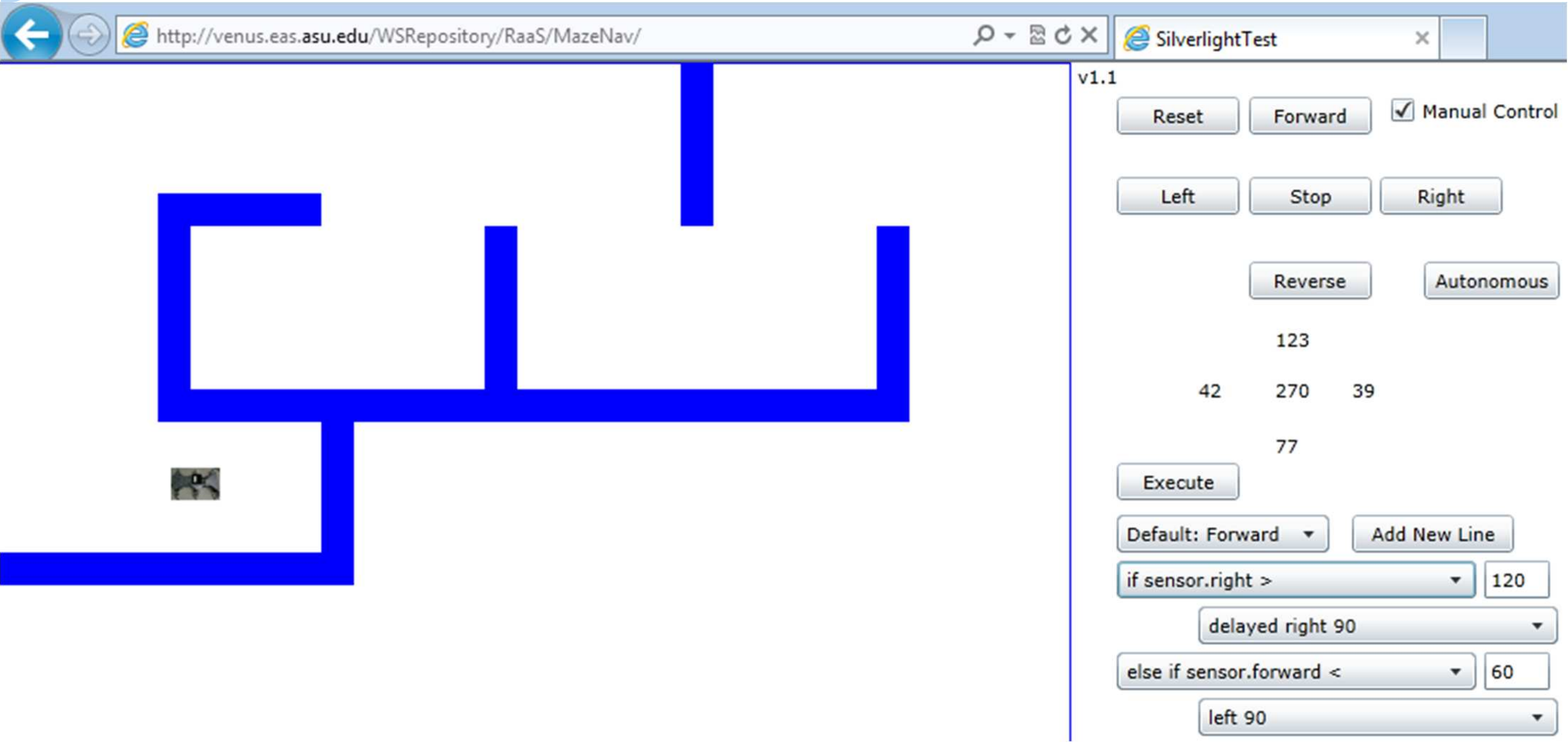
An **on-line** navigation problem: solving a maze from the inside.

An on-line algorithm receives its input gradually rather than all at once.

It must make decisions based on this partial input.

Online Programming of Wall Following Robot

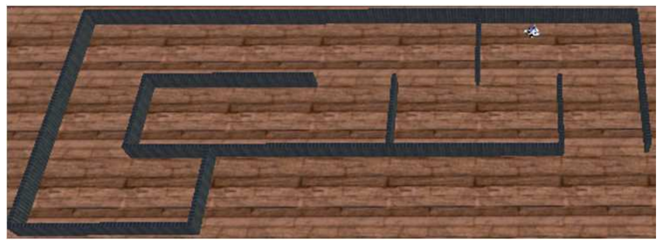
http://venus.eas.asu.edu/WSRepository/eRobotic/

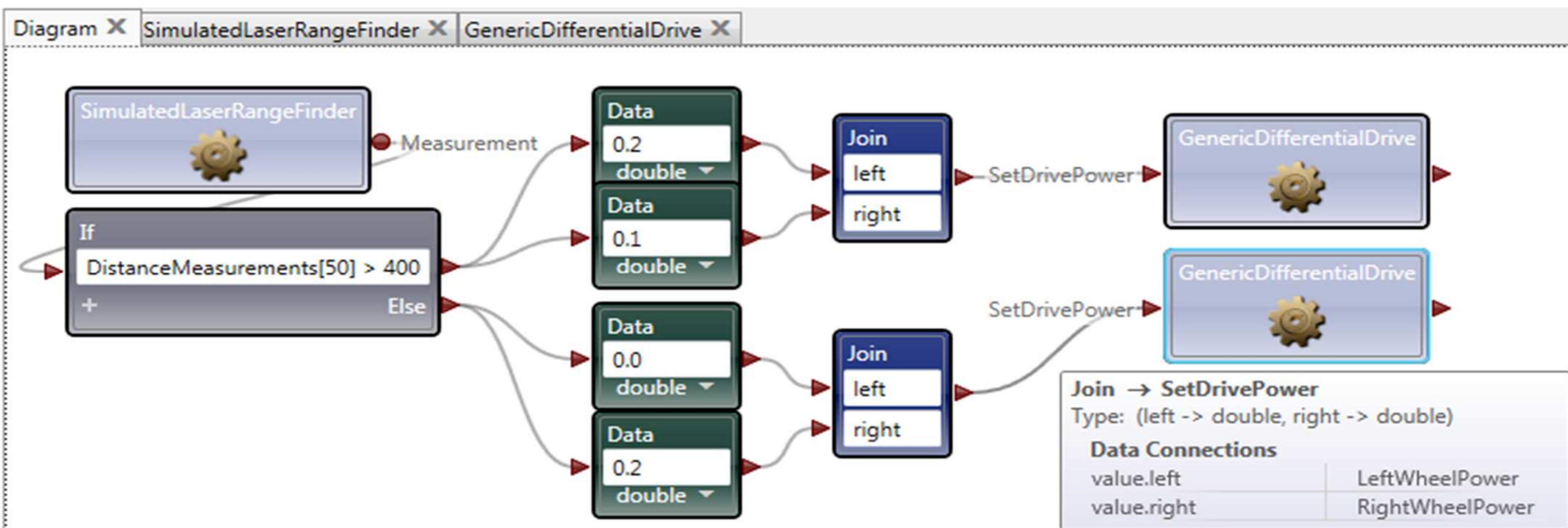


VPL Implementation

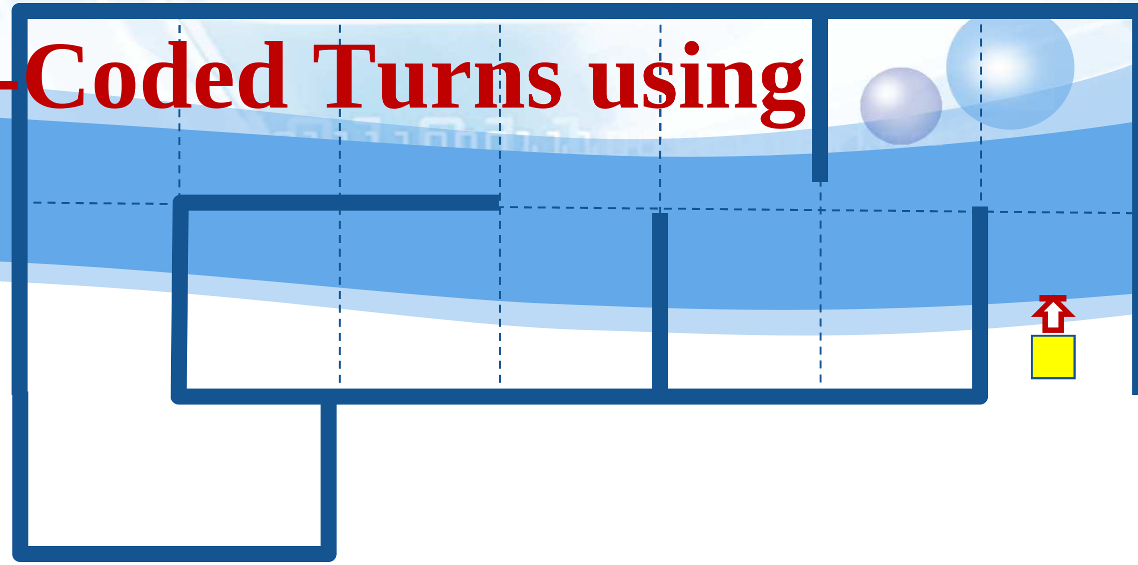
- Install ASU Maze into:

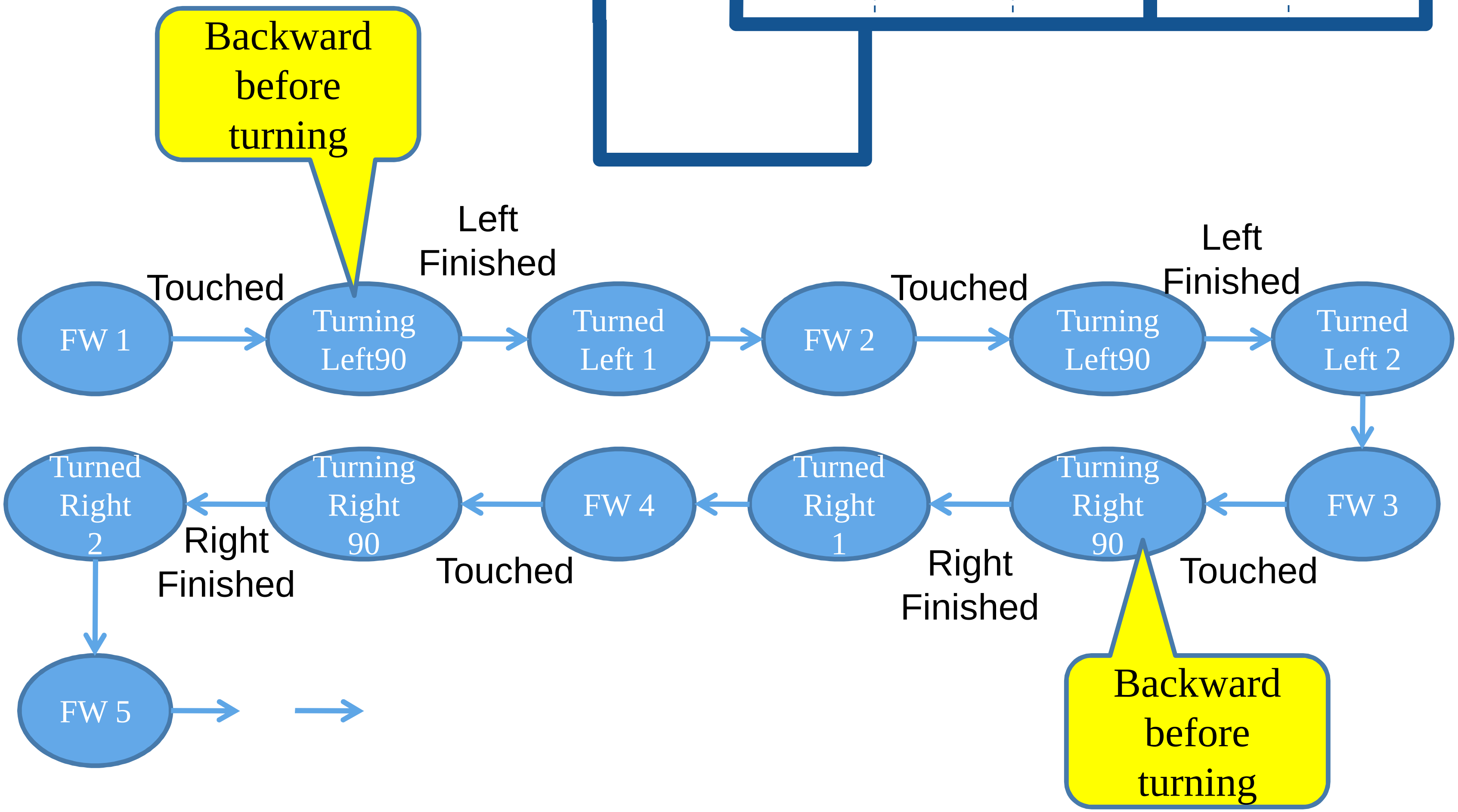
C/Documents and Settings/User/Microsoft Robotic Dev Studio 4/samples/Config



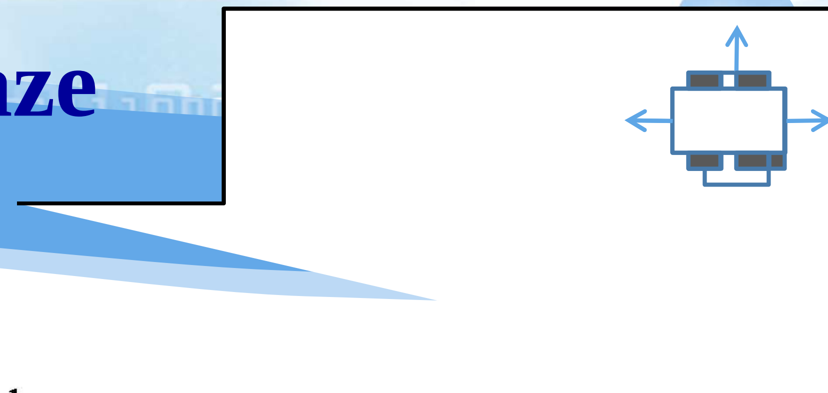


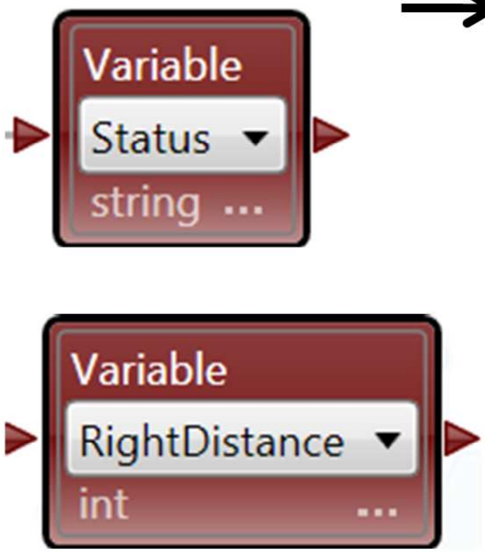
FSM for Hard-Coded Turns using Touch Sensor

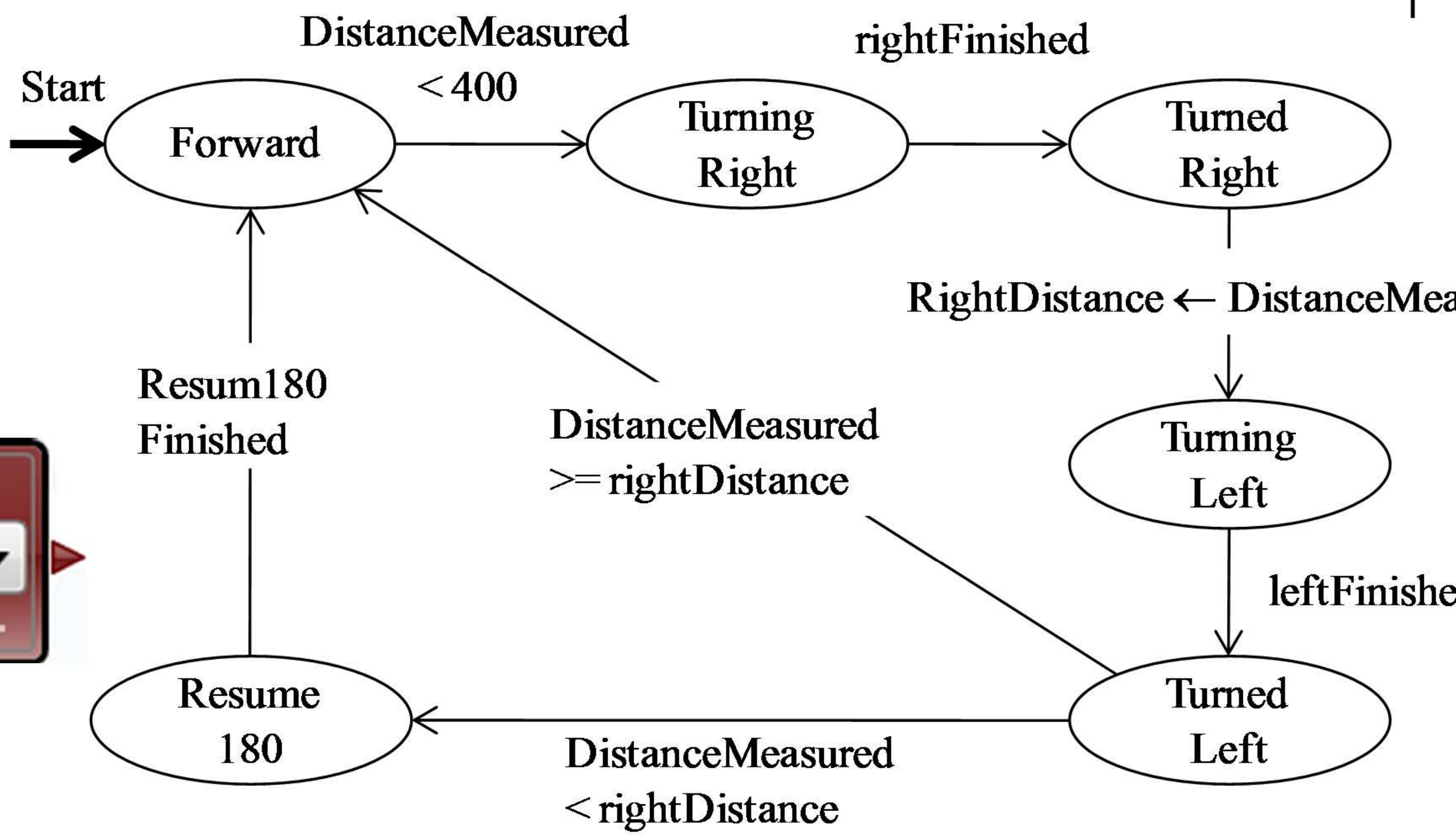




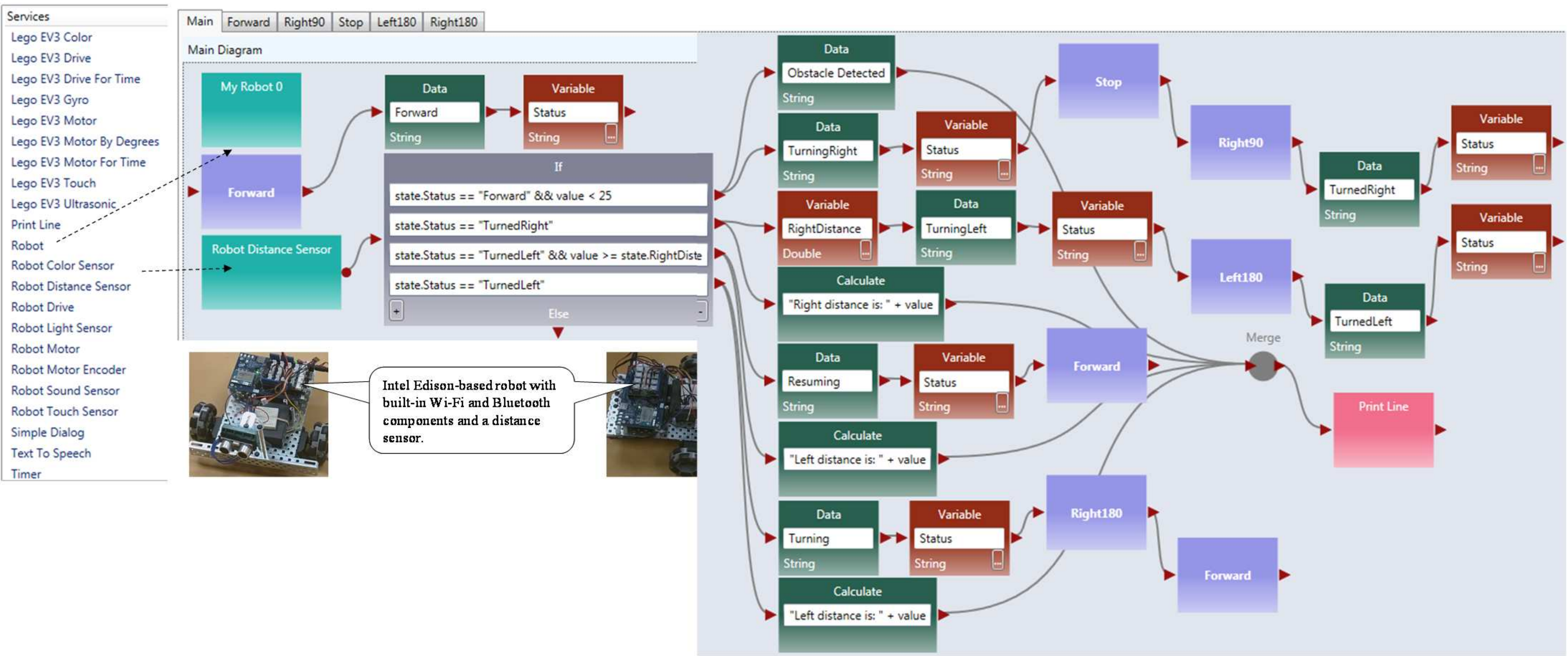
FSM of Robot in a Maze



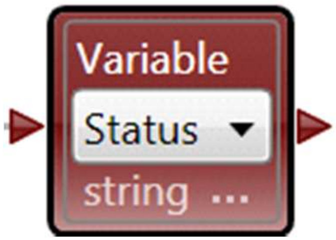


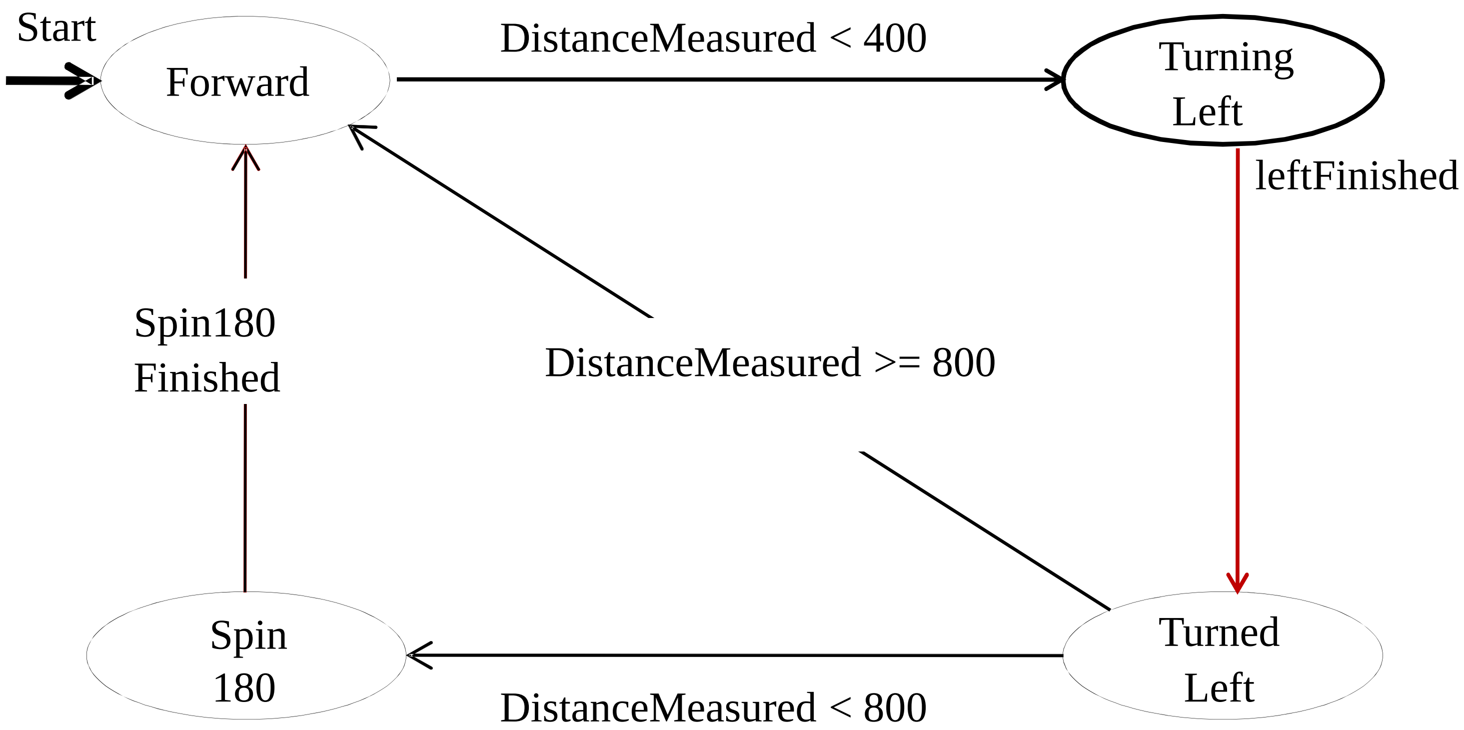


Implementation of the 'right-then-left' FSM

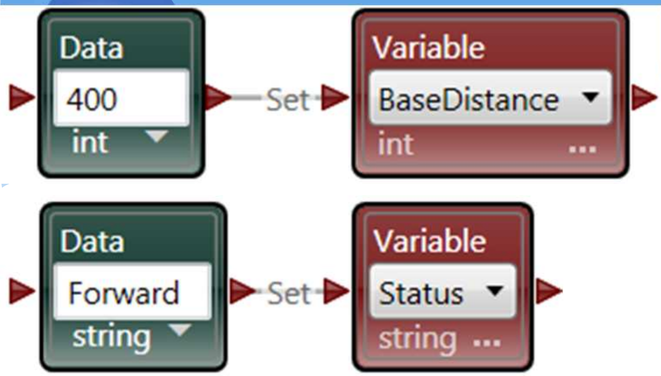


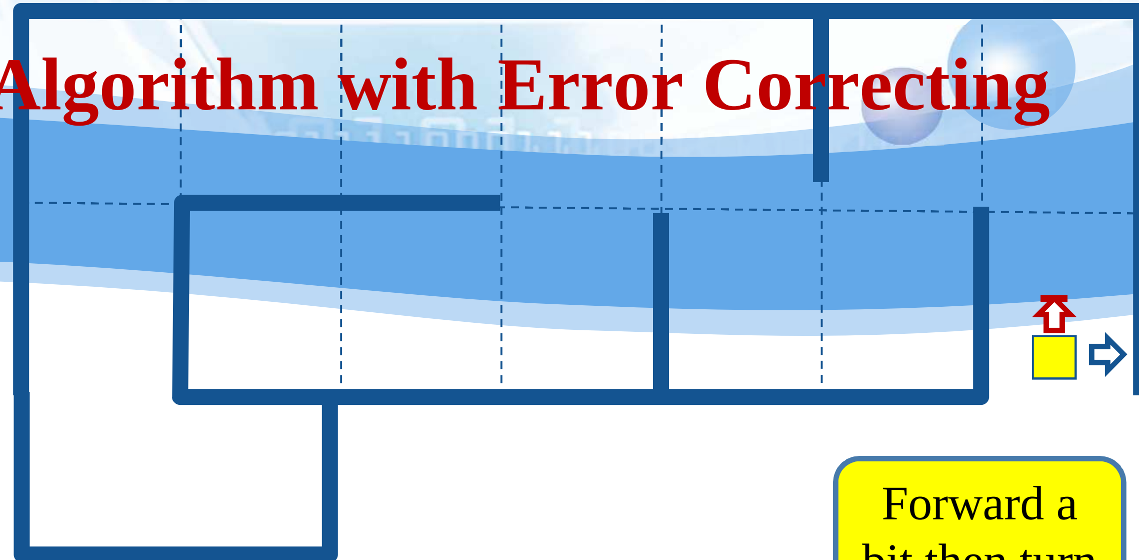
Greedy Algorithm based on the First Working Solution

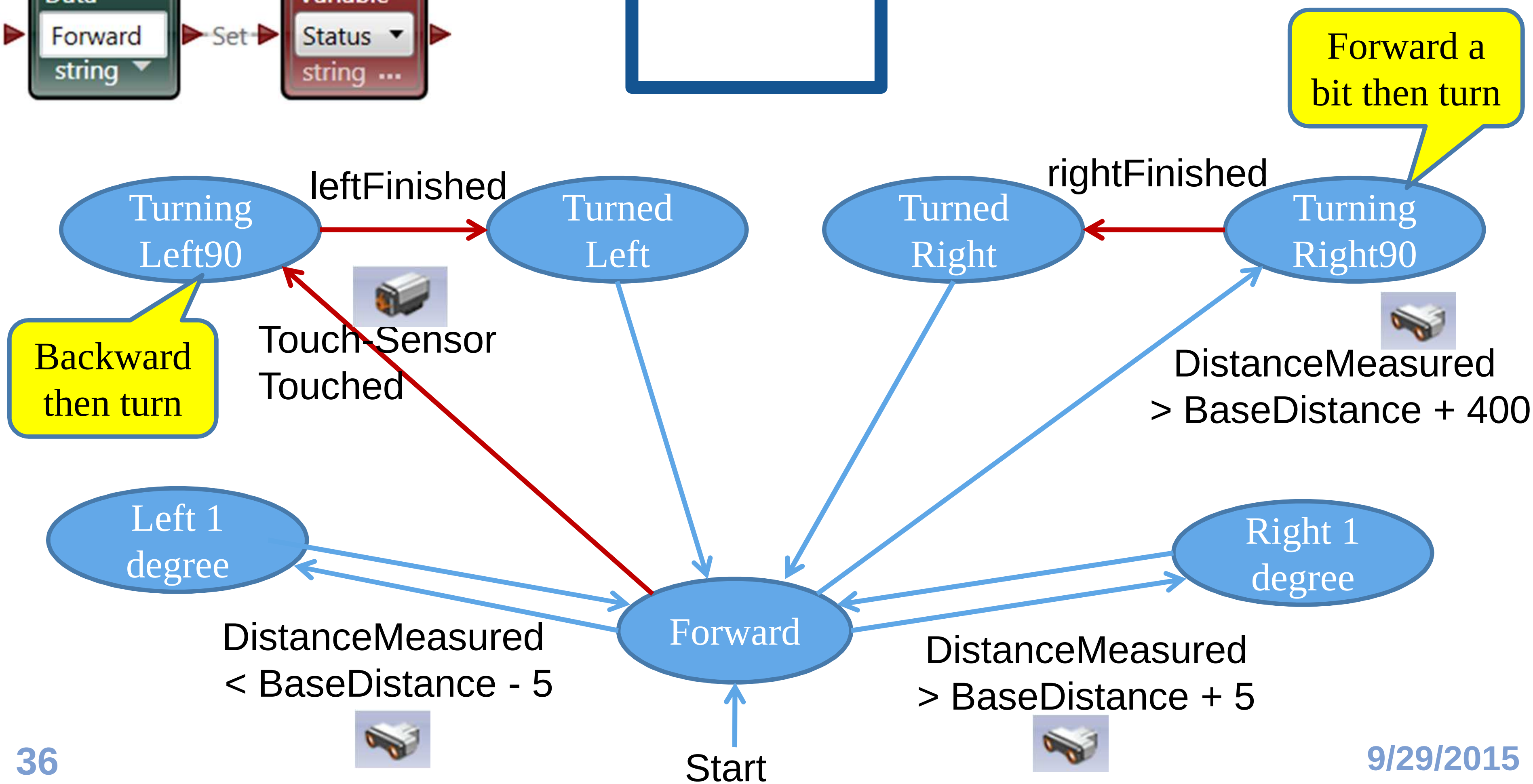




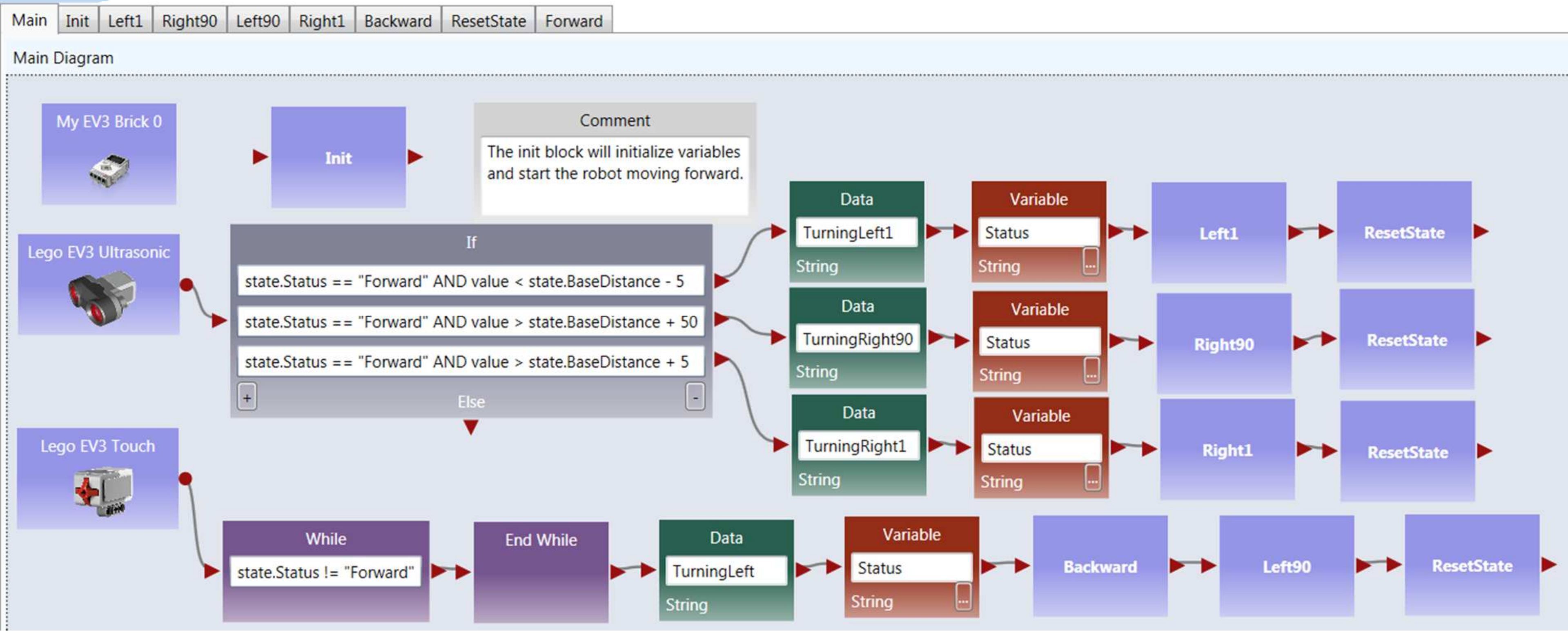
Right-Wall-Following Algorithm with Error Correcting



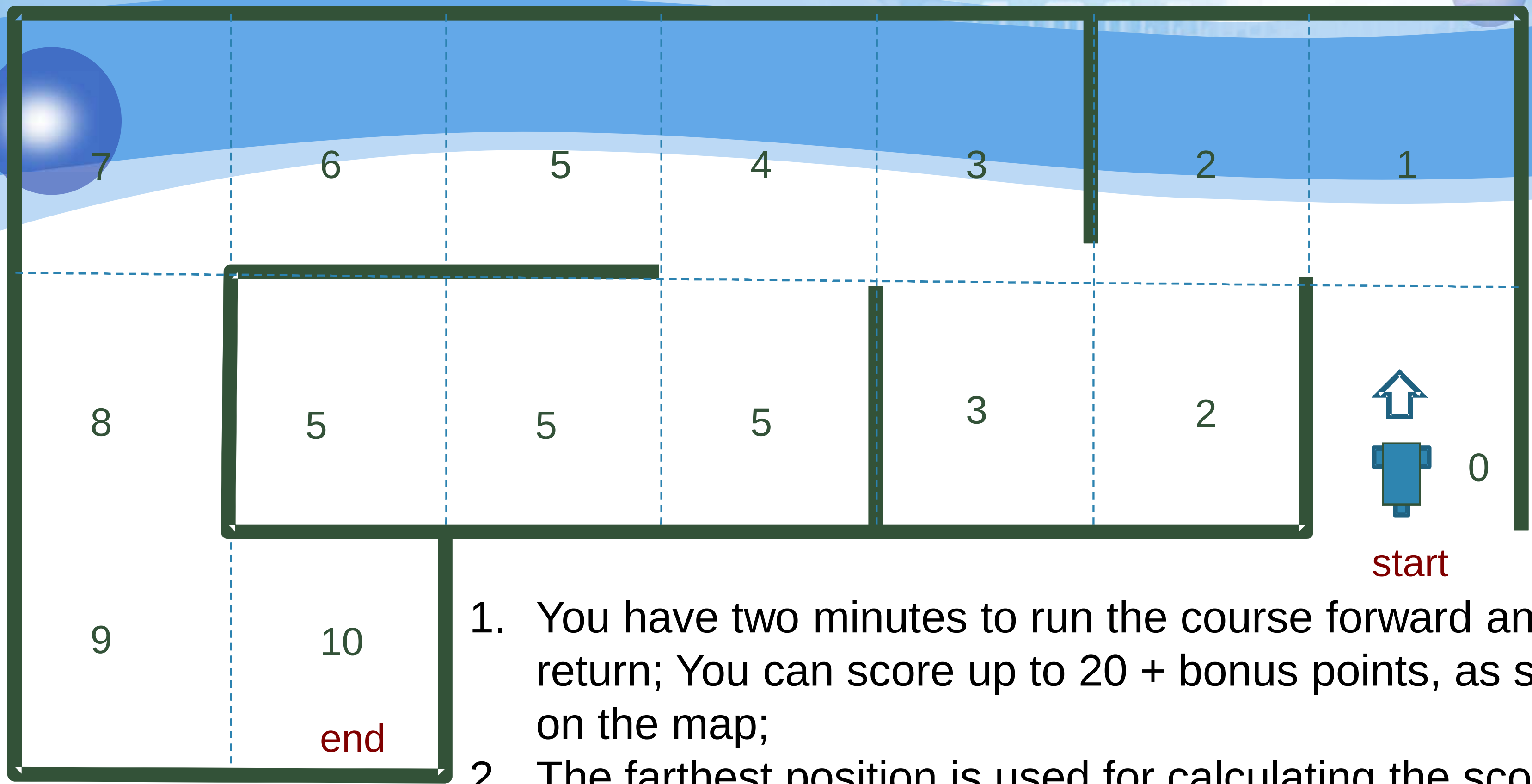




Wall-Following Diagram in VPL



Maze Navigation Game with Artificial Intelligence



1. You have two minutes to run the course forward and return; You can score up to 20 + bonus points, as shown on the map;
2. The farthest position is used for calculating the score;
3. If forwarding failed in the middle, you can take the robot to the end position to run the backward part;

Grading Scales:

4. If you use sensor to detect the front wall, + 10% bonus
5. If you use sensor(s) to detect front and side walls + 20% bonus points;
6. If you do not touch robot for the return trip, you receive 2 bonus points.