Phase 3

Performance Trials & Model-Based Testing

### Name: Alexander Habegger and Mohan Talluri

### Class: CSE 511 Model-Driven Software Engineering

### Professor: Dr. Eric Rapos

### Date: December 8, 2023

### 

### Table of Contents:

[Code & Charts Evidence 1](#_l4wk5qnex35c)

[StarUML Statechart 1 - Diagram & Robust Video 2](#_vnp7x7g5uj9v)

[StarUML Statechart 1 - Java Code 3](#_qobwpt1v30j)

[StarUML Statechart 2 - Diagram & Robust Video 6](#_5he9p6xvf2tj)

[StarUML Statechart 2 - Java Code 7](#_un9e9bkaudlk)

[StarUML Statechart 3 - Diagram & Robust Video 10](#_2u3t2b7n5rim)

[StarUML Statechart 3 - Java Code 11](#_7bzzmcyp6yqz)

[StarUML Statechart 3 - Symbolic Execution Tree 15](#_j26bvye2g133)

[StarUML Statechart 3 - Test Suite & Video Scenarios 16](#_2i30mnwovnju)

[StarUML Statechart 4 - Diagram & Robust Video 17](#_p2mhtz1j4nzp)

[StarUML Statechart 4 - Java Code 18](#_hnmhntx2qt6l)

[StarUML Statechart 4 - Symbolic Execution Tree 21](#_51j8psgbhv8i)

[StarUML Statechart 4 - Test Suite & Video Scenarios 22](#_qc96v1qwmgdn)

# 

# Code & Charts Evidence

| **Evidence** | **Link** |
| --- | --- |
| Code for UML2Lejos | <https://colab.research.google.com/drive/1ye7YJaQgTm9F6caRn8OZdJtrtzsCFJX8?usp=sharing> |
| Statechart MDJ Files | <https://drive.google.com/drive/folders/1B_G-aEU2GJiZH2fQj7wygBwdeNtkcI5H?usp=sharing> |
| Video Evidence | <https://drive.google.com/drive/folders/1-2PybGwFCQJlhwgfl6GbbXnBj_-q-ru_?usp=sharing> |

# 

# StarUML Statechart 1 - Diagram & Robust Video



| **Video Name** | **Link** | **Path Description** |
| --- | --- | --- |
| Chart 1 Robust 1 | <https://drive.google.com/file/d/1_WCFZJ_UVBeOGVqNDMOA-cgx_wlN9QsV/view?usp=drive_link> | 1. Idle for 1 second 2. Move backwards 3. Run over red paper 4. Rotate right 5. Move forwards 6. Encounter object closer than 0.3 meters 7. Stop |
| Chart 1 Robust 2 | <https://drive.google.com/file/d/1lZRkz4hqi6WTIj2xFw4RKhNRgoOr99EX/view?usp=drive_link> | 1. Idle for 1 second 2. Move backwards 3. Run over blue paper 4. Rotate left 5. Move forwards 6. Encounter object closer than 0.3 meters 7. Stop |

# StarUML Statechart 1 - Java Code

**import** lejos.hardware.motor.EV3LargeRegulatedMotor;

**import** lejos.hardware.port.MotorPort;

**import** lejos.hardware.port.SensorPort;

**import** lejos.robotics.Color;

**import** lejos.robotics.RegulatedMotor;

**import** lejos.robotics.SampleProvider;

**import** lejos.hardware.sensor.EV3UltrasonicSensor;

**import** lejos.hardware.sensor.EV3ColorSensor;

**import** lejos.hardware.Button;

**public** **class** Rover {

// Setup Motor Mechanisms

**public** **static** RegulatedMotor *m1* = **new** EV3LargeRegulatedMotor(MotorPort.***A***);

**public** **static** RegulatedMotor *m2* = **new** EV3LargeRegulatedMotor(MotorPort.***B***);

// Setup Proximity Sensing Mechanism

**public** **static** EV3UltrasonicSensor *u* = **new** EV3UltrasonicSensor(SensorPort.***S1***);

// Setup Color Sensing Mechanism

**public** **static** EV3ColorSensor *c* = **new** EV3ColorSensor(SensorPort.***S2***);

**public** **static** **void** main(String args[]) **throws** InterruptedException {

String state = "initial";

**while**(**true**) {

**if**(state == "initial"){

**while**(**true**){

state = "IDLE";

**break**;

}

}

**else** **if**(state == "final"){

**break**;

}

**else** **if**(state == "IDLE"){

*m1*.stop();

*m2*.stop();

**while**(**true**){

Thread.*sleep*(1000);

state = "BACKWARD";

**break**;

}

}

**else** **if**(state == "BACKWARD"){

*m1*.setSpeed(250);

*m2*.setSpeed(250);

**while**(**true**){

*m1*.backward();

*m2*.backward();

**if** (*getColor*() == Color.***RED***) {

state = "ROTATE\_RIGHT";

*m1*.stop();

*m2*.stop();

**break**;

}

**if** (*getColor*() == Color.***BLUE***) {

state = "ROTATE\_LEFT";

*m1*.stop();

*m2*.stop();

**break**;

}

}

}

**else** **if**(state == "FORWARD"){

*m1*.setSpeed(250);

*m2*.setSpeed(250);

**while**(**true**){

*m1*.forward();

*m2*.forward();

**if** (*getDistance*() < 0.3) {

state = "final";

*m1*.stop();

*m2*.stop();

**break**;

}

}

}

**else** **if**(state == "ROTATE\_LEFT"){

*m1*.setSpeed(450);

*m2*.setSpeed(250);

*m1*.backward();

*m2*.forward();

Thread.*sleep*(950);

*m1*.stop();

*m2*.stop();

**while**(**true**){

state = "FORWARD";

**break**;

}

}

**else** **if**(state == "ROTATE\_RIGHT"){

*m1*.setSpeed(250);

*m2*.setSpeed(450);

*m1*.forward();

*m2*.backward();

Thread.*sleep*(950);

*m1*.stop();

*m2*.stop();

**while**(**true**){

state = "FORWARD";

**break**;

}

}

}

}

/\*\*

\*

\* The method gives distance of proximity sensor

\*

\* **@return** the distance from an object in meters

\*/

**public** **static** **float** getDistance() {

**float** t\_distance = 0.0f;

**final** SampleProvider dist = *u*.getDistanceMode();

**float** [] sample = **new** **float**[dist.sampleSize()];

dist.fetchSample(sample, 0);

t\_distance = sample[0];

**return** t\_distance;

}

/\*\*

\* The method returns color of color sensor

\*

\* **@return** and integer value representing the detected color, matching the Color constants

\*/

**public** **static** **int** getColor() {

SampleProvider color = *c*.getColorIDMode();

**float**[] sample = **new** **float**[color.sampleSize()];

color.fetchSample(sample, 0);

**return** (**int**)sample[0];

}

}

# StarUML Statechart 2 - Diagram & Robust Video



| **Video Name** | **Link** | **Path Description** |
| --- | --- | --- |
| Chart 2 Robust | <https://drive.google.com/file/d/1xFCG8t1VrXFSEH5Y_8NoDWOdAsqJWQ3s/view?usp=drive_link> | 1. Idle 2. Press escape button 3. Move forwards 4. Encounter object closer than 0.1 meters 5. Move backward for 2 seconds 6. Rotate left 7. Move forwards 8. Run over blue paper 9. Idle 10. Press enter button 11. Stop |

# 

# StarUML Statechart 2 - Java Code

**import** lejos.hardware.motor.EV3LargeRegulatedMotor;

**import** lejos.hardware.port.MotorPort;

**import** lejos.hardware.port.SensorPort;

**import** lejos.robotics.Color;

**import** lejos.robotics.RegulatedMotor;

**import** lejos.robotics.SampleProvider;

**import** lejos.hardware.sensor.EV3UltrasonicSensor;

**import** lejos.hardware.sensor.EV3ColorSensor;

**import** lejos.hardware.Button;

**public** **class** Rover {

// Setup Motor Mechanisms

**public** **static** RegulatedMotor *m1* = **new** EV3LargeRegulatedMotor(MotorPort.***A***);

**public** **static** RegulatedMotor *m2* = **new** EV3LargeRegulatedMotor(MotorPort.***B***);

// Setup Proximity Sensing Mechanism

**public** **static** EV3UltrasonicSensor *u* = **new** EV3UltrasonicSensor(SensorPort.***S1***);

// Setup Color Sensing Mechanism

**public** **static** EV3ColorSensor *c* = **new** EV3ColorSensor(SensorPort.***S2***);

**public** **static** **void** main(String args[]) **throws** InterruptedException {

String state = "initial";

**while**(**true**) {

**if**(state == "initial"){

**while**(**true**){

state = "IDLE";

**break**;

}

}

**else** **if**(state == "final"){

**break**;

}

**else** **if**(state == "IDLE"){

*m1*.stop();

*m2*.stop();

**while**(**true**){

**if** (Button.***ENTER***.isDown()) {

state = "final";

**break**;

}

**if** (Button.***ESCAPE***.isDown()) {

state = "ROTATE\_RIGHT";

**break**;

}

}

}

**else** **if**(state == "BACKWARD"){

*m1*.setSpeed(250);

*m2*.setSpeed(250);

**while**(**true**){

*m1*.backward();

*m2*.backward();

Thread.*sleep*(2000);

state = "ROTATE\_LEFT";

*m1*.stop();

*m2*.stop();

**break**;

}

}

**else** **if**(state == "FORWARD"){

*m1*.setSpeed(250);

*m2*.setSpeed(250);

**while**(**true**){

*m1*.forward();

*m2*.forward();

**if** (*getDistance*() < 0.1) {

state = "BACKWARD";

*m1*.stop();

*m2*.stop();

**break**;

}

**if** (*getColor*() == Color.***BLUE***) {

state = "IDLE";

*m1*.stop();

*m2*.stop();

**break**;

}

}

}

**else** **if**(state == "ROTATE\_LEFT"){

*m1*.setSpeed(450);

*m2*.setSpeed(250);

*m1*.backward();

*m2*.forward();

Thread.*sleep*(950);

*m1*.stop();

*m2*.stop();

**while**(**true**){

state = "FORWARD";

**break**;

}

}

**else** **if**(state == "ROTATE\_RIGHT"){

*m1*.setSpeed(250);

*m2*.setSpeed(450);

*m1*.forward();

*m2*.backward();

Thread.*sleep*(950);

*m1*.stop();

*m2*.stop();

**while**(**true**){

state = "FORWARD";

**break**;

}

}

}

}

/\*\*

\*

\* The method gives distance of proximity sensor

\*

\* **@return** the distance from an object in meters

\*/

**public** **static** **float** getDistance() {

**float** t\_distance = 0.0f;

**final** SampleProvider dist = *u*.getDistanceMode();

**float** [] sample = **new** **float**[dist.sampleSize()];

dist.fetchSample(sample, 0);

t\_distance = sample[0];

**return** t\_distance;

}

/\*\*

\* The method returns color of color sensor

\*

\* **@return** and integer value representing the detected color, matching the Color constants

\*/

**public** **static** **int** getColor() {

SampleProvider color = *c*.getColorIDMode();

**float**[] sample = **new** **float**[color.sampleSize()];

color.fetchSample(sample, 0);

**return** (**int**)sample[0];

}

}

# StarUML Statechart 3 - Diagram & Robust Video



| **Video Name** | **Link** | **Path Description** |
| --- | --- | --- |
| Chart 3 Robust | <https://drive.google.com/file/d/1ZbhStOvlQJRN9lM5npRJ0mkC0nfSXvZ9/view?usp=drive_link> | 1. Idle 2. Press the enter button 3. Move backwards 4. Run over red paper 5. Idle 6. Press the escape button 7. Move forwards 8. Encounter object closer than 0.2 meters 9. Turn right 10. Stop |

# StarUML Statechart 3 - Java Code

**import** lejos.hardware.motor.EV3LargeRegulatedMotor;

**import** lejos.hardware.port.MotorPort;

**import** lejos.hardware.port.SensorPort;

**import** lejos.robotics.Color;

**import** lejos.robotics.RegulatedMotor;

**import** lejos.robotics.SampleProvider;

**import** lejos.hardware.sensor.EV3UltrasonicSensor;

**import** lejos.hardware.sensor.EV3ColorSensor;

**import** lejos.hardware.Button;

**public** **class** Rover {

// Setup Motor Mechanisms

**public** **static** RegulatedMotor *m1* = **new** EV3LargeRegulatedMotor(MotorPort.***A***);

**public** **static** RegulatedMotor *m2* = **new** EV3LargeRegulatedMotor(MotorPort.***B***);

// Setup Proximity Sensing Mechanism

**public** **static** EV3UltrasonicSensor *u* = **new** EV3UltrasonicSensor(SensorPort.***S1***);

// Setup Color Sensing Mechanism

**public** **static** EV3ColorSensor *c* = **new** EV3ColorSensor(SensorPort.***S2***);

**public** **static** **void** main(String args[]) **throws** InterruptedException {

String state = "initial";

**while**(**true**) {

**if**(state == "initial"){

**while**(**true**){

state = "IDLE";

**break**;

}

}

**else** **if**(state == "final"){

**break**;

}

**else** **if**(state == "IDLE"){

*m1*.stop();

*m2*.stop();

**while**(**true**){

**if** (Button.***ESCAPE***.isDown()) {

state = "FORWARD";

**break**;

}

**if** (Button.***ENTER***.isDown()) {

state = "BACKWARD";

**break**;

}

}

}

**else** **if**(state == "BACKWARD"){

*m1*.setSpeed(250);

*m2*.setSpeed(250);

**while**(**true**){

*m1*.backward();

*m2*.backward();

**if** (*getColor*() == Color.***RED***) {

state = "IDLE";

*m1*.stop();

*m2*.stop();

**break**;

}

**if** (*getColor*() == Color.***BLUE***) {

state = "ROTATE\_LEFT";

*m1*.stop();

*m2*.stop();

**break**;

}

**if** (*getColor*() == Color.***BLUE***) {

state = "ROTATE\_LEFT";

*m1*.stop();

*m2*.stop();

**break**;

}

}

}

**else** **if**(state == "FORWARD"){

*m1*.setSpeed(250);

*m2*.setSpeed(250);

**while**(**true**){

*m1*.forward();

*m2*.forward();

**if** (*getColor*() == Color.***BLUE***) {

state = "IDLE";

*m1*.stop();

*m2*.stop();

**break**;

}

**if** (*getDistance*() < 0.2) {

state = "ROTATE\_RIGHT";

*m1*.stop();

*m2*.stop();

**break**;

}

}

}

**else** **if**(state == "ROTATE\_LEFT"){

*m1*.setSpeed(450);

*m2*.setSpeed(250);

*m1*.backward();

*m2*.forward();

Thread.*sleep*(950);

*m1*.stop();

*m2*.stop();

**while**(**true**){

state = "final";

**break**;

}

}

**else** **if**(state == "ROTATE\_RIGHT"){

*m1*.setSpeed(250);

*m2*.setSpeed(450);

*m1*.forward();

*m2*.backward();

Thread.*sleep*(950);

*m1*.stop();

*m2*.stop();

**while**(**true**){

state = "final";

**break**;

}

}

}

}

/\*\*

\*

\* The method gives distance of proximity sensor

\*

\* **@return** the distance from an object in meters

\*/

**public** **static** **float** getDistance() {

**float** t\_distance = 0.0f;

**final** SampleProvider dist = *u*.getDistanceMode();

**float** [] sample = **new** **float**[dist.sampleSize()];

dist.fetchSample(sample, 0);

t\_distance = sample[0];

**return** t\_distance;

}

/\*\*

\* The method returns color of color sensor

\*

\* **@return** and integer value representing the detected color, matching the Color constants

\*/

**public** **static** **int** getColor() {

SampleProvider color = *c*.getColorIDMode();

**float**[] sample = **new** **float**[color.sampleSize()];

color.fetchSample(sample, 0);

**return** (**int**)sample[0];

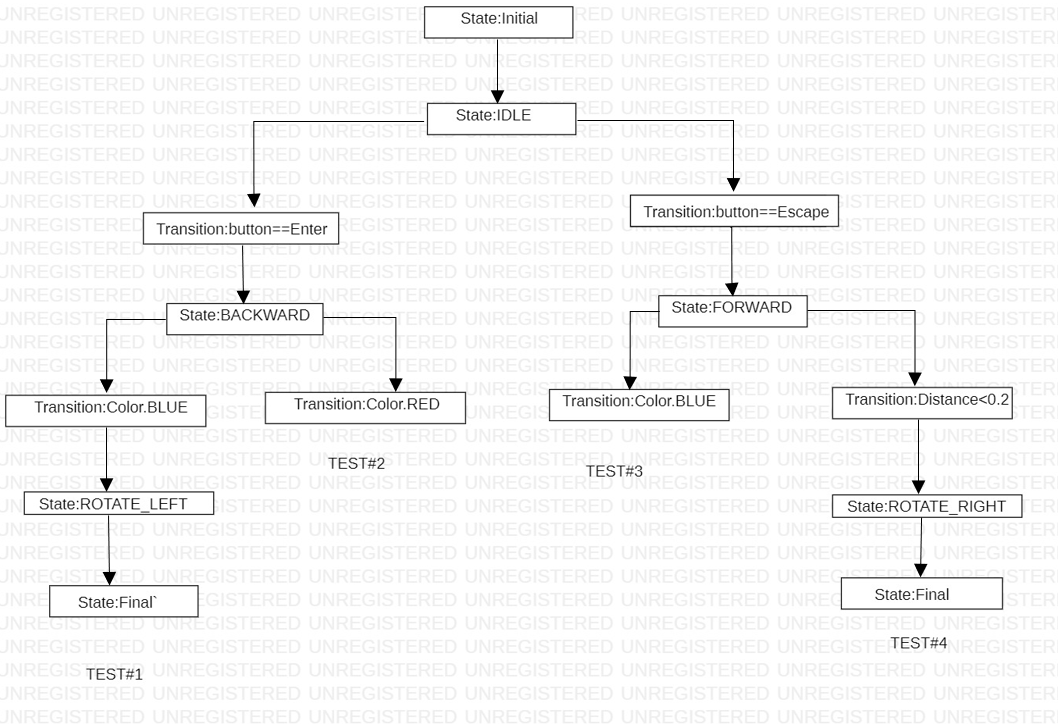
}

}

# 

# 

# StarUML Statechart 3 - Symbolic Execution Tree



By creating a symbolic execution tree for Statechart 3, we have determined that 4 testing branches provide system coverage.

# StarUML Statechart 3 - Test Suite & Video Scenarios

| **Test Case #1**   1. Hit the enter button 2. Robot will move backwards 3. Robot will run over blue paper 4. Robot will turn 90 degrees left 5. Robot will stop | **Test Case #2**   1. Hit the enter button 2. Robot will move backwards 3. Robot will run over red paper 4. Robot will idle |
| --- | --- |
| **Test Case #3**   1. Hit the escape button 2. Robot will move forwards 3. Robot will run over blue paper 4. Robot will idle | **Test Case #4**   1. Hit the escape button 2. Robot will move forwards 3. Robot will encounter object closer than 0.2 meters 4. Robot will turn 90 degrees right 5. Robot will stop |

| **Video Name** | **Link** | **Scenario** |
| --- | --- | --- |
| Chart 3 Test Case 1 | <https://drive.google.com/file/d/1-PMfddmGYGwOaRqGcP2rnCWieQH1KgZM/view?usp=drive_link> | Test Case #1 |
| Chart 3 Test Case 2 | <https://drive.google.com/file/d/1GRJG6Q5Na6Ci1-w8103KM2esEFwbHlMS/view?usp=drive_link> | Test Case #2 |
| Chart 3 Test Case 3 | <https://drive.google.com/file/d/14xz9qM1mpKbDl5_ji36oYGI5ikm5I9Yk/view?usp=drive_link> | Test Case #3 |
| Chart 3 Test Case 4 | <https://drive.google.com/file/d/1gZBLRoTvtAxio9DIHzUXmIBrWvokX-fH/view?usp=drive_link> | Test Case #4 |

# 

# StarUML Statechart 4 - Diagram & Robust Video



| **Video Name** | **Link** | **Path Description** |
| --- | --- | --- |
| Chart 4 Robust | <https://drive.google.com/file/d/150uWcU_2nVXdmZ9JiHsuBSYvngOrMhdg/view?usp=drive_link> | 1. Idle 2. Press the enter button 3. Move forwards 4. Run over red paper 5. Turn left 6. Move backward for 2 seconds 7. Idle 8. Press the enter button 9. Move forwards 10. Encounter an object less than 0.2 meters 11. Turn right 12. Move backwards for 2 seconds 13. Idle 14. Press the escape button 15. Stop |

# 

# StarUML Statechart 4 - Java Code

**import** lejos.hardware.motor.EV3LargeRegulatedMotor;

**import** lejos.hardware.port.MotorPort;

**import** lejos.hardware.port.SensorPort;

**import** lejos.robotics.Color;

**import** lejos.robotics.RegulatedMotor;

**import** lejos.robotics.SampleProvider;

**import** lejos.hardware.sensor.EV3UltrasonicSensor;

**import** lejos.hardware.sensor.EV3ColorSensor;

**import** lejos.hardware.Button;

**public** **class** Rover {

// Setup Motor Mechanisms

**public** **static** RegulatedMotor *m1* = **new** EV3LargeRegulatedMotor(MotorPort.***A***);

**public** **static** RegulatedMotor *m2* = **new** EV3LargeRegulatedMotor(MotorPort.***B***);

// Setup Proximity Sensing Mechanism

**public** **static** EV3UltrasonicSensor *u* = **new** EV3UltrasonicSensor(SensorPort.***S1***);

// Setup Color Sensing Mechanism

**public** **static** EV3ColorSensor *c* = **new** EV3ColorSensor(SensorPort.***S2***);

**public** **static** **void** main(String args[]) **throws** InterruptedException {

String state = "initial";

**while**(**true**) {

**if**(state == "initial"){

**while**(**true**){

state = "IDLE";

**break**;

}

}

**else** **if**(state == "final"){

**break**;

}

**else** **if**(state == "IDLE"){

*m1*.stop();

*m2*.stop();

**while**(**true**){

**if** (Button.***ENTER***.isDown()) {

state = "final";

**break**;

}

**if** (Button.***ESCAPE***.isDown()) {

state = "FORWARD";

**break**;

}

}

}

**else** **if**(state == "BACKWARD"){

*m1*.setSpeed(250);

*m2*.setSpeed(250);

**while**(**true**){

*m1*.backward();

*m2*.backward();

Thread.*sleep*(2000);

state = "IDLE";

*m1*.stop();

*m2*.stop();

**break**;

}

}

**else** **if**(state == "FORWARD"){

*m1*.setSpeed(250);

*m2*.setSpeed(250);

**while**(**true**){

*m1*.forward();

*m2*.forward();

**if** (*getDistance*() < 0.2) {

state = "ROTATE\_RIGHT";

*m1*.stop();

*m2*.stop();

**break**;

}

**if** (*getColor*() == Color.***RED***) {

state = "ROTATE\_LEFT";

*m1*.stop();

*m2*.stop();

**break**;

}

}

}

**else** **if**(state == "ROTATE\_LEFT"){

*m1*.setSpeed(450);

*m2*.setSpeed(250);

*m1*.backward();

*m2*.forward();

Thread.*sleep*(950);

*m1*.stop();

*m2*.stop();

**while**(**true**){

state = "BACKWARD";

**break**;

}

}

**else** **if**(state == "ROTATE\_RIGHT"){

*m1*.setSpeed(250);

*m2*.setSpeed(450);

*m1*.forward();

*m2*.backward();

Thread.*sleep*(950);

*m1*.stop();

*m2*.stop();

**while**(**true**){

state = "BACKWARD";

**break**;

}

}

}

}

/\*\*

\*

\* The method gives distance of proximity sensor

\*

\* **@return** the distance from an object in meters

\*/

**public** **static** **float** getDistance() {

**float** t\_distance = 0.0f;

**final** SampleProvider dist = *u*.getDistanceMode();

**float** [] sample = **new** **float**[dist.sampleSize()];

dist.fetchSample(sample, 0);

t\_distance = sample[0];

**return** t\_distance;

}

/\*\*

\* The method returns color of color sensor

\*

\* **@return** and integer value representing the detected color, matching the Color constants

\*/

**public** **static** **int** getColor() {

SampleProvider color = *c*.getColorIDMode();

**float**[] sample = **new** **float**[color.sampleSize()];

color.fetchSample(sample, 0);

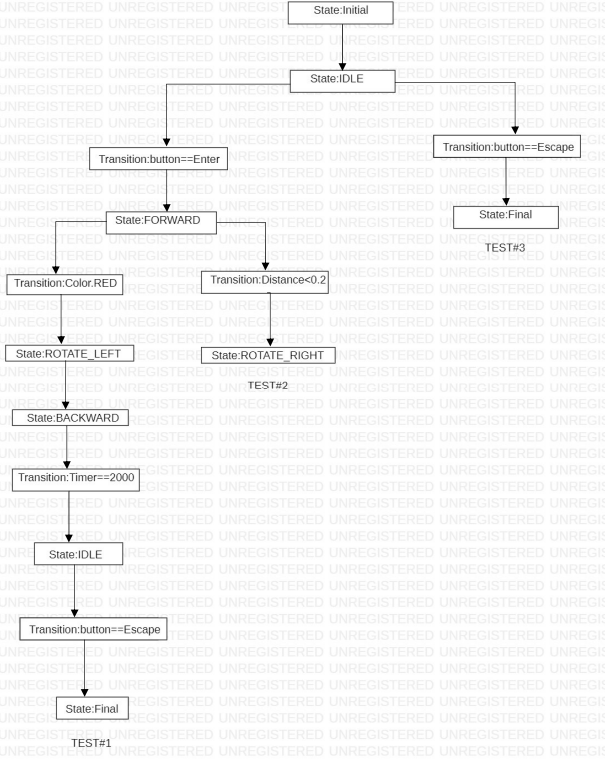
**return** (**int**)sample[0];

}

}

# 

# StarUML Statechart 4 - Symbolic Execution Tree



By creating a symbolic execution tree for Statechart 3, we have determined that 3 testing branches provide system coverage.

# StarUML Statechart 4 - Test Suite & Video Scenarios

| **Test Case #1**   1. Hit the enter button 2. Robot will move forwards 3. Robot will run over red paper 4. Robot will turn 90 degrees left 5. Robot will move backwards for 2 seconds 6. Robot will idle | **Test Case #2**   1. Hit the enter button 2. Robot will move forwards 3. Robot will encounter an object closer than 0.2 meters 4. Robot will turn 90 degrees right 5. Robot will start to move backwards |
| --- | --- |
| **Test Case #3**   1. Hit the escape button 2. Robot will stop |  |

| **Video Name** | **Link** | **Scenario** |
| --- | --- | --- |
| Chart 4 Test Case 1 | <https://drive.google.com/file/d/1zfZ5epjMcbBBXH7EQc2j5On1pUiVtTeq/view?usp=drive_link> | Test Case #1 |
| Chart 4 Test Case 2 | <https://drive.google.com/file/d/1_wH9JT8mgxb9dmBhjIGd4kDM_RcPpYIJ/view?usp=drive_link> | Test Case #2 |
| Chart 4 Test Case 3 | <https://drive.google.com/file/d/1Q20pWrQCoKfS-zBWmdeGDgpUb0O2s92S/view?usp=drive_link> | Test Case #3 |