Scala Notes:

Install Intellij then Scala and lastly SBT

Types –

Scala can create it’s own types

Everything in Scala has a type and easily catches errors at compile time

How to create a int function and call:

sum(2,2)

def sum (n1: Int, n2: Int): Int = n1 + n2

How to create a boolean function and call:

Println(lightSwitch(true))

def lightSwitch(switchOn: Boolean): String = if switchOn then “Lights On” else “Lights Off”

Proteus Notes (Adding RV to the Proteus Language) –

RV – Runtime Verification, HSM- Hierarchical state machines

Abstract

* The compiler is modified to produce programs so that RV can be added
* Add language constructs for RV which can be monitored and respond to events
* Make programs to make RV more convenient

Introduction

* Goal for proteus is to create a language that is safer for JPL system engineers to develop software for autonomous systems
* Proteus uses actors and Hierarchical state machines
* RV is a method for analyzing program behavior using a log generated during execution
* A monitor verifies that traces, sequence of events, satisfy certain properties or makes judgement about the program

Technology Used

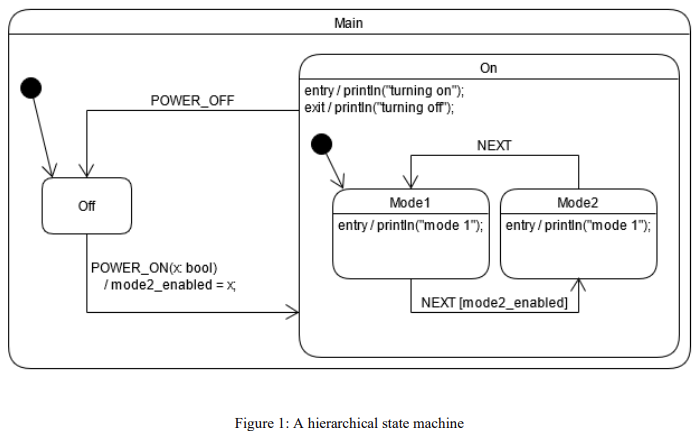
* Proteus compiler is written in C++ and it outputs C++ programs, also uses CMake for building the project, lcov for generating code coverage reports, Catch2 as a unit testing framework, and cxxopts  
  to parse command line options

Actors

* Actors are a model for systems in which each component is a independent entity
* They share no resources with each other, they carry out their own actions. But they are only able to affect each indirectly by messages
* The actor model uses the amount of concurrency which is obvious from the program structure and it is also safer with getting bugs

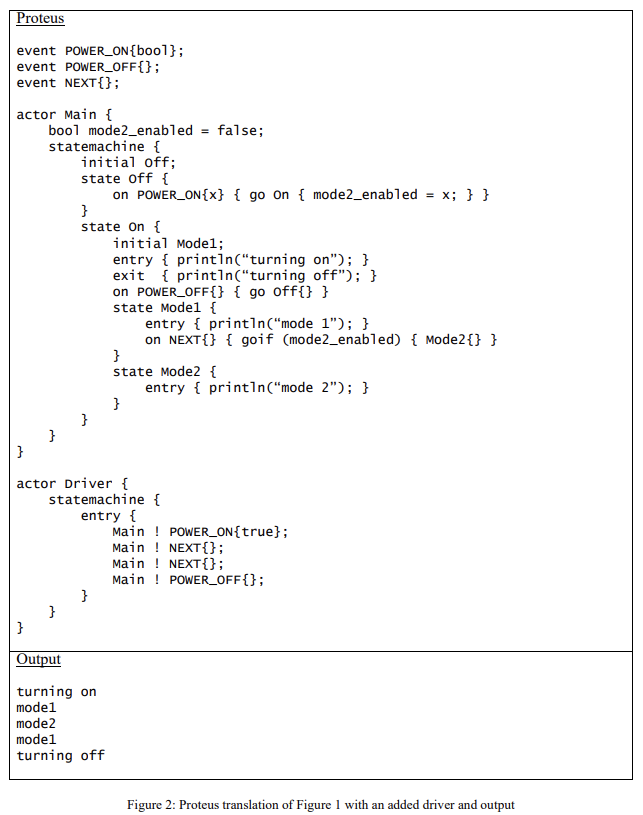
Hierarchical State Machines

* HSM are an extension of state machines to make them more suitable for modeling complex systems
* HSM can have HSM inside with the internal HSM’s inheriting the parent’s state



Proteus

* Proteus is based around actors and hierarchical state machines. A Proteus program consists, primarily, of definitions of actors, HSMs, and events. Each actor may have zero or one HSMs, and each HSM belongs to one actor. Events in Proteus are the messages that actors pass to each other as well as the events that HSMs respond to. Actors handle events by defining event handlers. These also perform state transitions.
* Actor given an HSM with *statemachine* keyword, Event handlers are introduced with the *on* keyword, Transitions are marked with *go* or *goif*



* Actors are static, they cannot be created or destroyed and exist for the duration of the program
* Static constructor can safely access the data of super states or actors
* Actors cannot access data of other actors
* Events behave like class definitions and must be initiated

Runtime Verification

Is a dynamic analysis method which checks whether a given run of a program is correct with respect to some specified properties

RV can be done online during or offline after execution

Runtime Library

(Towards a Systems Programming Language Designed for Hierarchical State Machines)