Implementing FSM, Statechart in Stateflow

Aarushi Ranjan, SIT, IIT Delhi

***Abstract*— A state refers to the status or an encoding of the history of all previous conditions that are responsible for changes in the present reaction of the system, at a particular point in time. A finite state machine is a mathematical model for the computation of an abstract specific task. Also known as the abstract state machine or finite state automata, it possesses a finite number of states as the name suggests, out of which it can be in only one in a given instant of time. In FSMs, a particular input can cause a change in the state of the machine, also known as transition, and can be leveraged to simulate sequential logic. Extending the concept of FSMs, a state chart is a diagrammatic representation used to model the reactive behaviour of a system by describing its states. State charts are of various types and can be executes in MATLAB's Stateflow.In this report, attempts have been made to represent two problems stopwatch and traffic control using state diagrams in Stateflow.**

***Index Terms*—Finite state machines, State diagrams, Stateflow**

I. Introduction

Statecharts are standard formalisms that help represent a system consisting of its finite states, out of which the system can be only in any one given state in an instant of time. Beefing up the concept of finite state machines, statecharts use higraphs, which are diagrammatic objects helping formalize relation between states of a system into a structure. These graphs are different than traditional mathematical graphs in the sense that they bring a notion of depth and orthogonality in the system, and can also be hierarchical.Since most modern systems consist complex designs owing to a number of factors such as hardware, software and data complexity, it can be said that they demonstrate event driven behaviours. An event is said to have occurred when a system reacts to an input, changes its internal state and cause a change in the output of the system. For example, during a racing game, the 'up' arrow key pressed by a user, here input, causes the change in state of the vehicle from being stationary to moving forward. The output observed is the forward movement of the vehicle on the screen. This is also an example of a reactive system, where the system is reacting to the input coming from the environment, such as the keyboard input from the user. Similarly, complex systems are also autonomous and timed. While autonomous systems are those that can attain a given set of goals in an environment that is changing continuously, through observing several states, and without the intervention of any human, timed systems are those that model the timing behaviour of systems and are an extension of a set of real clocks. To model and validate the behaviour of such complex reactive, autonomous and timed systems, state diagrams are used.

State diagrams follow the syntax consisting states and transitions, which can be combined as per the requirements. State charts can be hierarchical or composite, and orthogonal or concurrent. A hierarchical state chart refers to the diagram where the principle of hierarchy followed for a system is reflected in its states, indicating the presence of superstate comprising of other states. For entities whose behaviours are described using orthogonal state, a concurrent state chart is used.

The rest of the paper is as follows: Section II describes gives a technical introduction to the problem and our approach, Section III describes the method employed to implement the approach, and Section IV comprises the results and insights from the simulations. Finally Section V concludes the report.

II. Theory

A. Understanding the Stopwatch Problem

A digital stopwatch is a circuit or a timepiece that measures the actual amount of time elapsed specifically between its periods of activation and deactivation. Simply, it measures the time interval of an event once a stimuli is provided to it. This means that it works by pressing a start button and stops when the stop button is pressed. It consists of a face where the time data is recorded and displayed, along with buttons that stop/start the time, and reset it, if required. To initiate the functioning of a stopwatch, the start button is pressed. Upon completion of the event, the measurement of time interval in a stopwatch stops. It can display the time in minutes, seconds, or even clock pulses. An ideal stopwatch can display the count form 0 to 59, for a 60s time interval.

The motivation for using state chart for the stopwatch arises from representing the complex functionalities and states of the stopwatch in easier manner as compared to traditional state transition systems which increase in complexity due to their flat nature. In the case of a stopwatch, it is also important to represent the orthogonality of the entities occurring in independent behaviours, but reacting to the same event such as the extended implemented problems of lapping and storage. Lastly, it is quite easy to check and validate the functioning of the model using state charts.

For constructing the state chart, careful considerations are given to: count minutes and seconds, starting from 00:00 to 59:59 and then repeat counting; control of the stopwatch by two commands, start' and 'lap', allowing to take lap times through freezing the recording; and finally, a switch for memory storage that stores the current value of the stop watch in the temporary memory.

B. Understanding the Traffic Lights Problem

Traffic lights are devices for signaling vehicle regulation messages on the road. Positioned at the intersection of roads or pedestrian crossings, they aim at regulating the traffic. These lights abide by a universal code for colour which provide the viewers with the right way to proceed on the road, lit up by

three LEDs within them. These lights are Red, Green and Yellow. The choice of this universal colour coding draws from the concepts and definitions in physics. While Red indicates the prohibition of all kinds of traffic, ordering vehicles to immediately stop, yellow indicates a warning, asking vehicles to slow down in order to prepare for an upcoming red light. Finally, the green light indicates movement and allows the traffic to start or continue moving in their way. These are also linked to safety, where red refer to danger or stop, and green refers to safe commuting for the traffic. A phase is often referred to as the duration for which a traffic light stays in a particular colour. The motivation for using state chart for the traffic light problem lies in decreasing the complexity of understanding states and their transitions since a number of requirement considerations exist; orthogonality of independent events reacting to the same behaviour; and much easier trace-ability from specification to design for the system. For constructing the traffic signal state chart several considerations are made: three different lights are coloured red, yellow and green; one light switches on at any instant of time; initial state of the system starts from red light; the cycle of lights is from red, green, yellow; the red light turns on for 60s duration, green for 55s and yellow for 5s duration respectively; existence of an interrupt by the police, hindering the autonomy; timer for displaying the time interval left for the current signalling light; the pedestrian can push a button to request for a green light, which will stay for atleast 60s, after which the transition to yellow takes place.

III. Implementation

The pillars of a state chart model are the states of the system and the transitions that occur on account of input variations. In traditional system development, there are several artifacts created that help describe the process and govern actions. They are created for every action taking place and are also received as inputs. However, they obey the state chart's syntax and are also responsible for defining transformations between states. In this implementation, first the requirements of the system are defined, leading to an initial model, after which it is simulated and tested. Upon manual checking the validation of fulfillment of these requirements are made.

A. Stopwatch Problem

The representation of the stopwatch is primarily done using counter variables for minutes and seconds. Fig 1 represents the stateflow diagram for the specification of the stopwatch problems described in Section II (A). The initial state of the system is Stopped. An event run triggers that starts the stopwatch. At this instant, the value of minutes and seconds both are initialized from 0. This is also done for the state to not pick any arbitrary garbage from the memory. Upon the occurrence of the event, the state of the system changes to started where there is a counter. A process of self-transition is occurring on the counter, which is implementing an if else case. The if condition, stated as work in the diagram checks for the value of seconds to be less than 59, after which the counter for seconds increases by 1. In the else condition, the if the seconds become equal to 59, the counter for minute increases. This is intuitive since as the second hand turns 60, it begins a new minute. Before it, the seconds keep on incrementing. Since no further provision to count hours is provided, the stopwatch must enter the stop state. When the functionality for lapping is added, the counters for minutes and seconds are initialized from zero and input variables min and sec are realized. The counter gets its value from the difference of the minute or second value and the counter itself. This describes the initial state of the system. In the lapping state, self transitions take place over an if else event. The event lapping checks if the seconds are greater than or equal to the counter, post which the increment in the counter takes place, both for the minutes and seconds. Similarly, in the else case, is the value of seconds is less than the counter for seconds, the minutes counter gets its value incrementally from the difference of minutes, counter value with further difference of value 1. The additional difference of 1 is not necessary for the seconds counter. Fig 2 and 3 represent implementation of the lapping function where two buttons start and lap are used for variables seconds and minutes. The restart is a state where the counters stop proceeding further. Fig 4 represents the stateflow for the integrated store/recall or memory function in a stopwatch. It stores the current value of the watch in a temporarily memory upon switching it on. When the switch is off, the stopwatch is reset to the collected value by the memory. The process is occurring in the memory, which explains the invisibility of any new number on the simulink. Finally, Fig 5 represents the comprehensive stateflow diagram for the problem.

B. Traffic Lights Problem

The problem of traffic lights is represented through three states red, yellow and green. While the states do not posses any grammatical properties corresponding to the actual traffic lights. There are several state variables defined which are kept track by the system. The transition for the system are defined keeping in mind the considerations from Section II (B).The artefacts of the system which are extended to the interface comprise of input events and output events. The output events would comprise of a showcase of the lights, which means the lights actually lighting up. The implementation made is completely autonomous, as in it doesn't accept any stimuli from its surrounding environment. Composite modeling has been used in state charts due to the fact that police interruptions can occur in the middle of the processes, leading to two states namely the normal and interrupted. Just like a state chart needs to be precisely in one default state, the composite states follow the similar rule. This also reduces the probability of non-determinism in systems. It is common for non-determinism to occur in complex systems. For example, in this problem of traffic lights, when the lights are switching from one colour to another, and a police office interrupts in between, there can be a state of non-determinism. A crucial component of the formulation of the state chart is the component of history. In this example, the history state is kept inside the composite state. The main function of this state is to remember the

value of the present state of the composite state, after the composite state has ended upon undergoing transition. For the case where the pedestrians press a button or request a green light to cross the road, four states are assumed. While three of these highlights the traffic lights, the fourth is a pending state with a self transition running upon the incremental condition of the system variable count. As earlier the system starts from Red state where the counter starts running from zero upon the trigger of the event. The transitions from red to green and pending to yellow check the counter value for greater than or equal to 60, because the minimum duration or phase of these states in 60seconds. After a check for count value less than 60 at the self transition green state, the transition from green to pending checks for the condition where an and operation is used for pedestrian and count against a counter of less than 60, which increments by 1 in the true case. Finally, a transition from green to yellow is marked once again for the condition where the pedestrian and count are less than 60, incrementing by a null value. The output variables are defined as sigR, sigG and sigY, for each of the lights.

IV. Results

A. Stopwatch

The state chart of different requirements under the stopwatch problem were modeled and run using the simulink. A pulse generators were used for the model to generate square wave pulses at regular intervals. At both the inputs of the stopwatch and memory function state chart, separate constant blocks with manual switches were lighted. Two displays connected to the output displayed the value of seconds and minutes upon running the simulation for a fixed time. Test cases for 100, 1000 and 5000 time pulses were observed. For the lapping part, a separate state chart was implemented, connected to two boolean valued constants and screens recording minutes and seconds. Additionally, scopes were planted at places to validate the signals in the system.

B. Traffic Lights

The state chart for the traffic lights problem was modeled and also run using the simulink. In a similar fashion to the above problem, the state chart model is tested by defining the trace of inputs and the probable output events. When transitions take place from one state to another in composite states, the first state is excited, while the second is entered.

V. Conclusions

Statecharts are a suitable formalism to represent a system that is autonomous, timed as well as reactive. While traditional modeling approaches find it hard to model non-deterministic systems, state charts present a suitable solution by helping in the easier definition of states, transitions, events and actions. The level of discrete abstraction offered through the encoded dynamics of the system, historical states and much simpler inclusion and representation of hierarchies provides a suitable motivation for its use across multiple genres of problem solving. Additionally, Simulink is a quick and user friendly tool helping access all the necessary components for circuit design and system modeling.

Appendix A
State Charts and Stateflow Diagrams for Given Problems

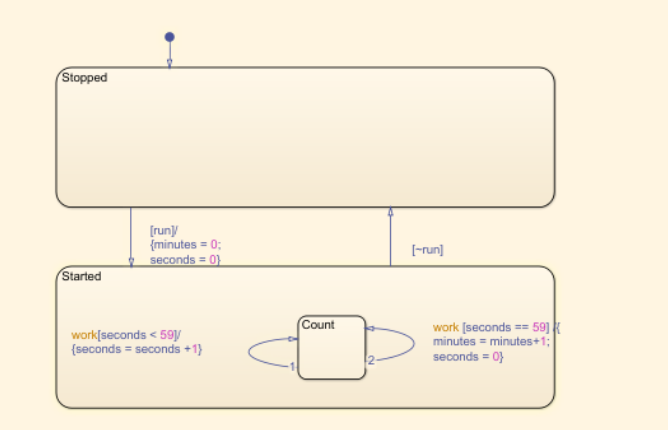

Fig. 1. A simple stopwatch state chart

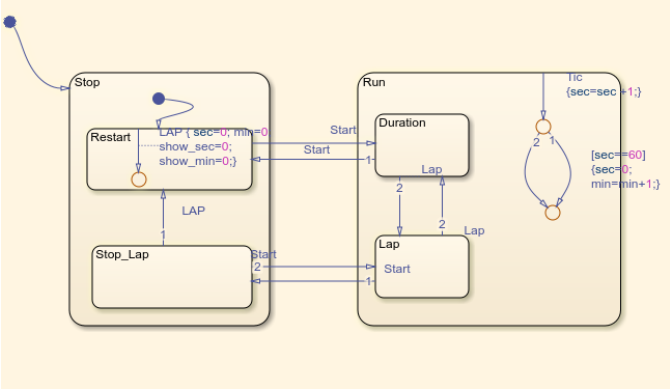

Fig. 2. Statechart for lapping feature (Method 1)

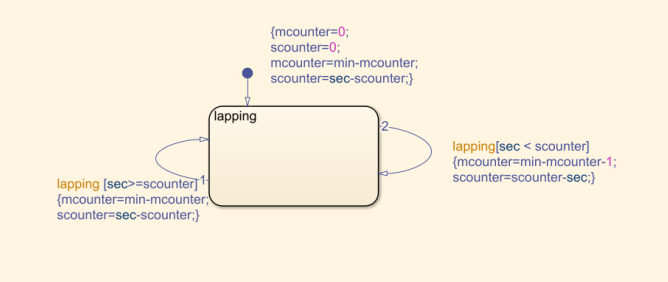

Fig. 3. State chart for lapping feature (Method2)

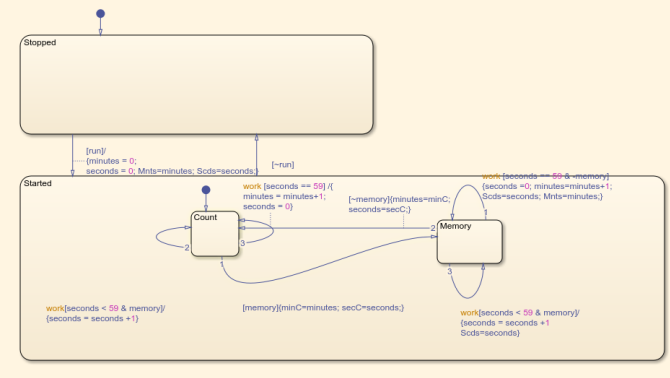

Fig. 4. Storage or Recall memory in the stopwatch that captures the current value and stores it in the memory.

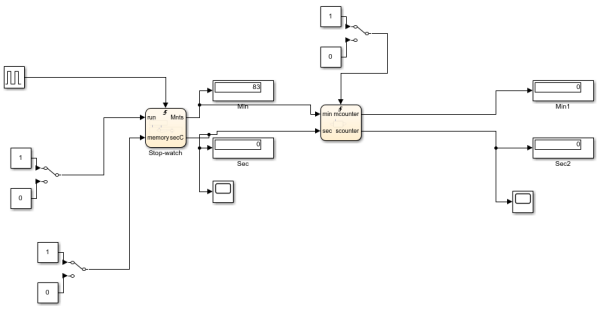

Fig. 5. Stateflow diagram for the entire sub problems for Stopwatch problem. The first state chart demonstrates the functionality of a simple stopwatch along with a storage/recall invisible memory functioning.

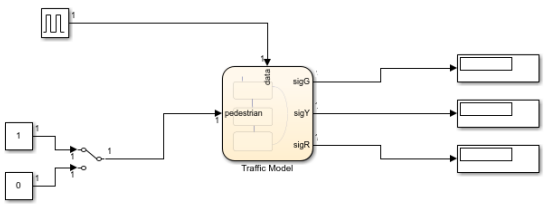

Fig. 7. State flow diagram for a three state traffic signal problem with the pedestrian constrained introduced.

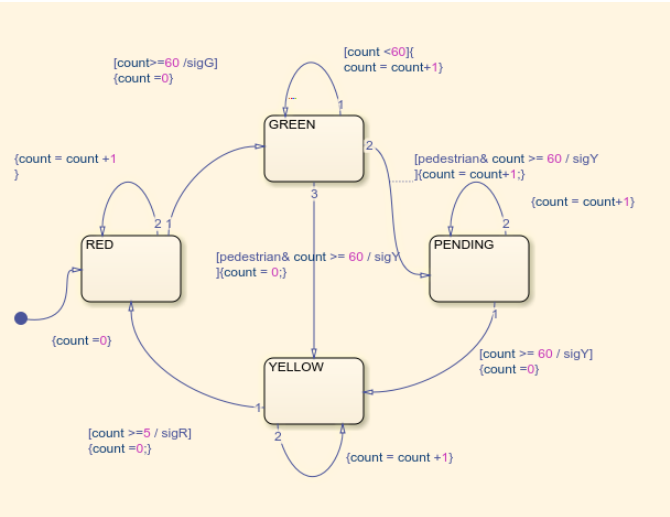

Fig. 6. State chart for a three state traffic signal problem with the pedestrian constrained introduced.