Translation of Simulink / Stateflow Models to HyTech

Debjit Pal
Co-Ordinated Science Laboratory
University of Illinois Urbana-Champaign
Urbana, Illinois 61801
Email: dpal2@illinois.edu

***Abstract*—Simulink / Stateflow (SLSF) has become a very standard tool now-a-days for the modeling of any complex system including automotive vehicles, manufacturing plants and complex controllers. But in the SLSF, the designer can only simulate the system for a limited number of inputs like any other simulation tools. This does not guarantee that the system will behave as desired in the worst case scenario also. Hence, there is a need to formally analyze the system to ensure that the system behavior never enters a *bad state* of operation. There are several tools available from academia like HyTech, PHAver and SpaceEx which can formally analyze the behavior of the systems by finding the reach set of the continuous variables of the system. In this work, we propose a systematic methodology to translate SLSF model to HyTech model which is amenable for formal analysis. The work is not complete and the tool implementation is in its way. In this report, first we define three well defined methods of modeling in Simulink / Stateflow. Then we describe our proposed method for the translation of SLSF model to HyTech model file.**

I. Introduction

Matlab Simulink-Stateflow [1] has now become an industry standard tool for modeling complex systems. It is widely used for modeling controllers of the plants, automotive vehicles, flights etc. The rich functional library of Simulink allows to model very complex system with appropriate accuracy in less time. Added with this, the capacity of StateFlow charts bring the notion of discrete states which are very natural to any kind of controllers. But Simulink / Stateflow is not sufficient to verify the system as it can check system's behavior on some execution but not all possible executions. Hence to ensure that the designed system never enters the *bad state*, there is a need to translate the SLSF models systematically to some format which is amenable for formal analysis. In literature there exists three prominent tools for doing such safety analysis of systems with linear dynamics by computing reach sets of the continuous variables of the systems. They are HyTech [2], PHAver [3] and SpaceEx [4]. There exists attempts by the researchers [7] where they have demonstrated a translation mechanism for the standalone Simulink or Stateflow models to HyTech and UPPAAL [5]. But there are several shortcomings of the proposed approach. For example, the method cannot translate combined SLSF model and they also consider a very small subset of the Simulink functional blocks for the translation which further inhibits the expressiveness of modeling. In this work we propose a technique where combined SLSF models can be translated for formal analysis. The rest of the report is arranged as follows : Section II describes three well defined modeling style in SLSF along with the merits and demerits. Section III describes the existing methodology briefly and in Section IV we describe our proposed method to augment the existing translation mechanism. Section V concludes this report.

II. Different Modeling Styles in Simulink/Stateflow

In this section we define three distinct well defined modeling styles in the SLSF. We will use an example of Two Thermostats from [7] as a running example in this section. Many systems like automotive vehicle, controllers, plants which are modeled in the SLSF has a continuous evolution part and a discrete state transition part. These two aspects can be modeled in SLSF in the following two ways :

1) Both continuous evolution and discrete transition are modeled in Stateflow Chart.
2) The discrete transition is modeled in the StateFlow Chart whereas the continuous evolution is modeled in the Simulink.

Keeping this in view, we present the following modeling formalisms in the following two sub sections.

A. Complete Modeling in StateFlow

In this modeling framework, both continuous evolution and the discrete state transitions depending upon the value of the continuous variables are modeled in the StateFlow chart. We explain this with the example of the Figure 1. The aim of this thermostat is to keep a room tempaerature within certain limits by alternately switching off and switching on one of the thermostat. We note that the Thermostat1 has two states namely *On1* and *Off1* and similar is the case for Thermostat2. The continuous evolutions are modeled as simple ODEs (like $x1\_dot = -0.02 * x1$) in the state itself. The Thermostat1 changes its mode from *On* to *Off* when the temperature goes beyond 70 units and switches from *Off* to *On* when the temperature goes below 40 units. The thresholds for similar switching for the Thermostat2 has been set at 65 units and 15 units respectively. We note that there is no means of communication between these two thermostats and they operate on their own i.e. they are free moving. In this method, the "update method" for the continuous variables like $x1$ is *continuous*. But this modeling method has several drawbacks. It does not allow to use the rich function library of Simulink with which modeling can be more expressive and accurate. Also, *continuous* update method [6] does not allow use of

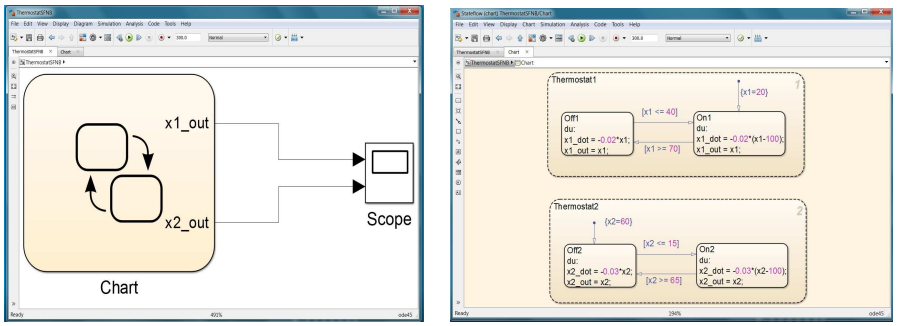

Fig. 1. Stateflow Model of a Simple Thermostat

Broadcast Synchronization label between the transitions of the different state machines in the StateFlow, use of temporal formulas in the state or on the transition. Hence, this modeling style is suitable for simple and small systems.

B. Simulink / Stateflow Combined Modeling

In this modeling framework, we model the continuous evolutions in the Simulink diagram and discrete transitions in the Stateflow charts. We explain the idea with the help the example of the Figure 2. All the differential equations of the different modes of operations of the two thermostats are modeled in the Simulink. Depending upon the value of the continuous variables, the discrete mode transition occurs in the StateFlow chart and the corresponding mode signals (like $x1Mode$ and $x2Mode$) are sent to the Simulink diagram. Depending upon the mode signal received from the Stateflow chart, the differential equations are altered by the *Switch* blocks. We can use rich set of functions available in the Simulink library to model very complex functions easily. Two points need to be noted here :

1) As soon as we separate the continuous evolution and the discrete transitions in Simulink and Stateflow respectively, "the update method" for the StateFlow chart need to be changed from *continuous* to *discrete* and a finite value of the sampling time has to be provided. This can be done via the *Model Explorer* available in the top panel of StateFlow chart.
2) Since all calculations are done in continuous mode in the Simulink and the update method in the Stateflow is discrete, hence we need to place a "Rate Transition" block at the interface of the StateFlow and Simulink for proper handling of signals. Without that, Matlab will report an error regarding the rate signal sampling rate and will abort the simulation.

With this modeling style we have decoupled the flow equations and the discrete transitions but still the modes of the state machines are free moving and they have no means of communication. But this decoupling comes with the advantage that now we can use the *synchronization* label across the transitions of the state machines which was not allowed earlier. The synchronization allows the state machines to communicate with each other and operate in an interleaved fashion. Next we explain how we can use the *synchronization* label in the modeling.

We explain the use of the sync label with the help of the example in the Figure 3. The sync labels are used on the transitions of the state machines. In the state machine Thermostat1, the transition from the Off1 state to On1 state is labeled as *[x1 <= 40]{send(OnOff,Thermostat2)}*. [x1<=40] is the guard condition for this transition. The *send* construct broadcasts the event of interest to other transitions of the other state machine. The first entity namely *OnOff* is the event to be broadcast and the second entity namely *ThermoStat2* is the state machine to which the event is broadcasted to. When the temperature falls below 40 units, the transition from *Off1* to *On1* is enabled and this transition is taken with the broadcast of the *OnOff* event. At the same time, Thermostat2 which was in On2 state, will have its transition from *On2* state to *Off2* state enabled due to the *OnOff* event. Similar things occur when the temperature goes above 70 units, the Thermostat1 switches from *On1* to *Off1* mode by broadcasting the *OffOn* event and the ThermoStat2 switches from *Off2* to *On2* mode as its transition is enabled by the broadcast event. Now the two state machines can communicate via the state label and they are no longer free moving. This kind of modeling is very helpful to model hybrid system where there are both continuous evolution and discrete transitions involved and the discrete transitions communicate via sync labels. For example, the famous Rail-Gate controller example from [8] has three interacting automata, one for each of the train, gate and controller. Each of them have continuous variables and discrete transitions. Depending upon the position of the train with respect to the gate, different transitions are possible. These transitions are synced with the transitions of the gate and controller automata with the help of the label namely *app*, *exit*, *lower* and *upper*. The presented modeling style is a natural framework to model these kind of systems very easily.

In the next section, we describe the existing methodology in brief for the translation of Stateflow Chart HyTech.

III. Existing Method of Translation from SLSF to HyTech

In this section we briefly explain the SLSF to HyTech translation methodology demonstrated in [7]. Hierarchical state machines can be converted into a flat StateFlow model by the process of *flattening*. A flat StateFlow chart does not have any sub-modes for any of its modes. A hierarchy tree htree = (V,E) is constructed with the set of vertices V and edges E such that $v \in V$ denotes a mode in the system and has and *unique id* and *type* associated with it. The *type* can be

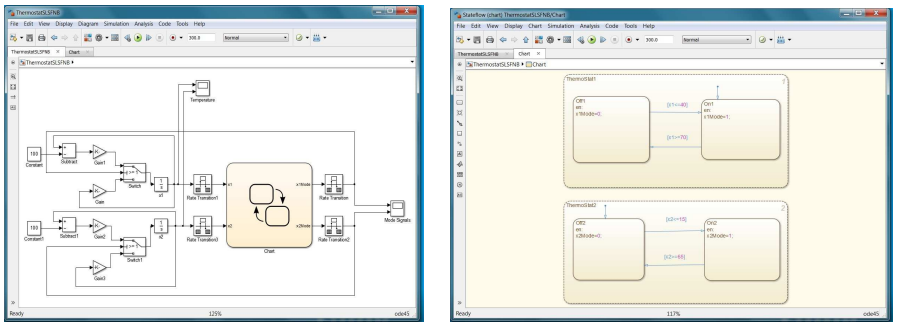

Fig. 2. Stateflow Model of a Simple Thermostat without Sync Label

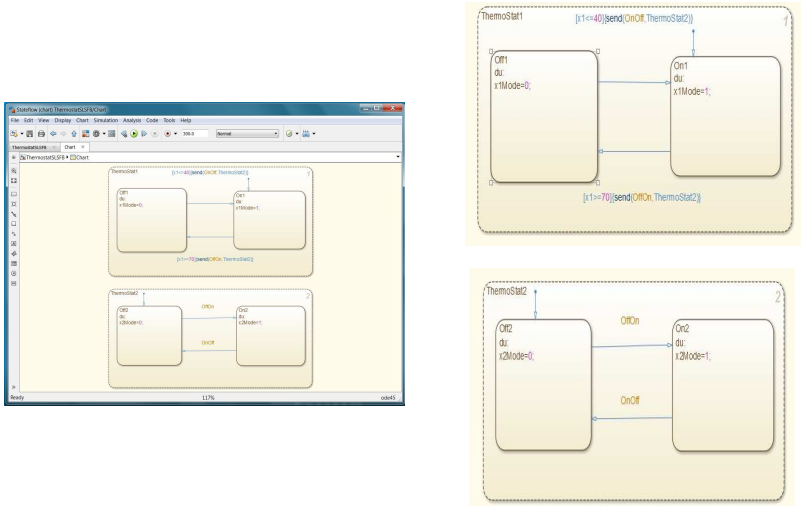

Fig. 3. Stateflow Model of a Simple Thermostat with Sync Label

and or *or*. each edge $e \in E$ connects vertices m and n if and only if n is a sub-mode of m. A configuration*c* of a StateFlow chart can be uniquely mapped to a subgraph of the hierarchy tree. A subgraph in which (i) the root node is included, (ii) for every *and* node which is included all of its children are included, and (iii) for every *or* node which is included one of its children is included is a valid configuration of the system. We explain the idea with the help of Figure 4 in the context of Figure 1. In Figure 4, node 7 is the *and* node and hence both of its children i.e. node 5 and 6 corresponding to the Thermostat1 and ThermoStat2 respectively are included. Now node 5 and node 6 both of them are *or* types i.e at a time, one of the sub-modes in each of these node can be active i.e. In Thermostat1 either *Off1* or *On1* can be active at a time. Similar is the case for ThermoStat2. Hence, the possible valid configurations for the Two Thermostat system of Figure 1 are 7.5.6.1.3, 7.5.6.2.3, 7.5.6.1.4 and 7.5.6.2.4. Basically it is the product of the states of all the state machines. Hence the HyTech automaton model will contain four states namely Thermostat1.ThermoStat2.Off1.Off2, Thermostat1.ThermoStat2.Off1.On2, Thermostat1.ThermoStat2.On1.Off2, Thermostat1.ThermoStat2.On1.On2.

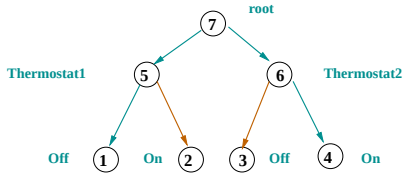

Fig. 4. Hierarchy of States in Thermostat Stateflow Model

IV. Proposed Method of Translation of Combined SLSF Model

Clearly the above mentioned methodology of taking product of the state machines will not scale if the model contains very large number of interacting state machines. Although creating the HyTech model will not be an issue but HyTech may fail to analyze such huge state machine. Here we leverage the power of the sync label. This can be done in the following two ways :

1) We can systematically prune out some of the state product combinations which are not possible due to the conflicting sync labels. That will reduce the total number of states in the resulting HyTech model. But still it may explode if the number of state machines are large.

2) We leverage the fact that HyTech and other tools support sync labels across the state machine to analyze interacting state machines. In this process, we do not take explicit product of the states of the state machines rather we translate each individual state machine to a automaton model with the sync label.

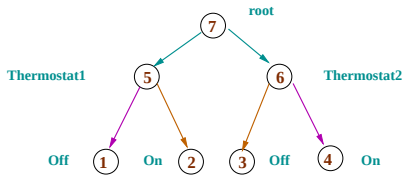

Fig. 5. Hierarchy of States in Thermostat Stateflow Model with Sync Label

With the option 1 as mentioned above the effective states in the HyTech model reduces to two as shown in Figure 5. The only legal configurations are 7.5.6.1.4 and 7.5.6.2.3 i.e. the violet edges form a valid state and brown edge form another valid state. We generate the guard condition and the reset condition of the resulting automata following the same method as proposed in [7]. But we are more interested in implementing the method as proposed in option 2 above which will help us to accommodate large state machines also. We propose the following steps for the translation :

1) We extract out the Simulink part from the combined model and generate the differential equations following the method described in [7]. Each of the differential equation will correspond to a value of the discrete mode signals which are output from Stateflow and and input to Simulink model.
2) We create an individual hybrid automaton model [8] for each of the state machine in the StateFlow chart including the guard and invariant conditions following the method described in [7]. In this intermediate hybrid automaton model, there will be no differential equation involved, rather they will contain the name of output of the Simulink model for that particular mode. This names will act as the flag to find out the appropriate differential equation for a particular mode from the Simulink model created in previous step.
3) In this final step, we substitute the correct differential equations in the hybrid automaton model created from StateFlow from the model created from Simulink by a *string matching*. This will give the actual model of the automaton that can be analyzed in HyTech. The proposed tool flow is shown in the Figure 6.

V. Conclusion

In this work we have proposed three different modeling paradigm in the SLSF and also explained its advantages and disadvantages. We have shown that decoupling the flow equations and the discrete transitions can help us to model the system more efficiently using more advanced and rich constructs of Simulink and StateFlow. Using sync label, we can model hybrid system in a natural way. We have proposed a systematic approach to extract HyTech models from combined Simulink / Stateflow model. The implementation is not

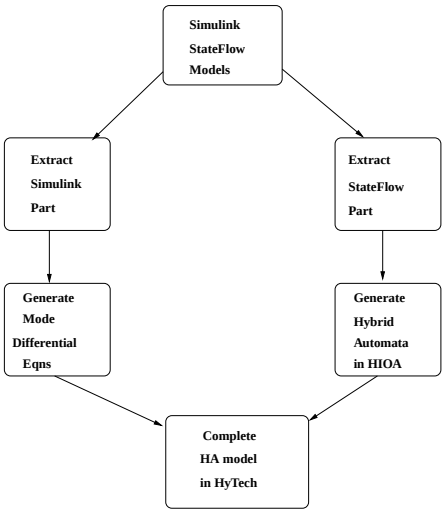

Fig. 6. Flowchart for the Translation Mechanism

complete and in process. In future, we also wish to support hierarchical Simulink models and more Simulink blocks so that richer class of SLSF models can be translated into HyTech model for formal analysis.

References

[1] Simulink-StateFlow Modeling http://www.mathworks.com/products/stateflow/
[2] Hytech http://embedded.eecs.berkeley.edu/research/hytech/
[3] PHAver http://www-verimag.imag.fr/~frehse/phaver_web/
[4] SpaceEx http://spaceex.imag.fr/
[5] UPPAAL http://www.uppaal.org/
[6] StateFlow Chart Update Method http://www.mathworks.com/help/stateflow/ug/setting-the-stateflow-block-update-method.html
[7] Translation of Simulink-StateFlow Models to Hybrid Automata, Karthik Manamcheri Sukumar, Master's Thesis, University of Illinois Urbana-Champaign, 2011.
[8] The Theory of Hybrid Automata, Thomas A. Henzinger, LICS, 1996, 278-292, IEEE Computer Society.