Hierarchy of Extended Finite-State Machines

Michal Soucha

August 14, 2017

1 Why use extended finite-state machines?

Extended finite-state machines (EFSM) are powerful abstract models of any system that can be described by a *regular language*. As all tasks of the General AI Challenge are regular languages, EFSMs can represent the tasks accurately and so provide correct outputs. The main advantage over other representations is that EFSMs are **easy to understand and analyse** for both people and computers, see the examples of EFSMs in Figure 1 on page 3.

An EFSM consists of finite number of states, transitions between them, and structures holding data. As only the deterministic behaviour is considered, the modelled system is always in exactly one state called the current state. The current state is changed according to inputs applied to the system and corresponding transitions. Each transition is labelled with an *input* symbol, an *output* symbol, a *reward*, a *guard* and an *action* such that the guard is a condition that needs to be satisfied to follow the transition, and the action can change the data structures. The guard and the action can be missing on a transition as can be seen in Figure 1. There can be several transition with the same input from a state but then the outputs, the rewards, or the guards need to distinguish the transitions so that at most one transition corresponds to the applied input and the observed output and reward. This secures that the system cannot be in several states at once. On the other hand, there can be no suitable transition from the current state, then the model does not describe the system. For example, when the task is changed, the model of the previous task does not represent the new task. EFSMs without the data structures and guards and actions on transitions are called *observable deterministic finite-state machines* (ODFSM) and they are accurate to model the traces, that is, the observed sequences of input-output-reward tuples.

2 Learning EFSM hierarchy

EFSMs modelling the tasks of Challenge contains just one cycle of transitions that start and ends in the initial state. The cycle correspond to one run of a task instance. This important property allows one to establish a hierarchy of EFSMs with a tree representation such that children are specializations of their parent. The root of hierarchy is an EFSM with just the initial state and requires an output on any applied input such that the reward can be arbitrary. Its child, a specialized model, can be for example an 1-state EFSM prescribing a particular output for all inputs (Micro-task 1). The root is thus a generalization of all other EFSMs in the hierarchy. Part of the hierarchy is captured in Figure 1.

Learning a new EFSMs, and so solving a task, has three phases as follows.

1. *Specialization* of the conjectured model – the conjectured model is initialized with the model of hierarchy that is consistent with the current trace but no its child is and it was used correctly the most amongst such models. Its specialization with the ODFSM covering the current trace finds common control pattern in the task instance like there is a command name followed by a colon (Micro-task 6 and up) or there is a period after the feedback.

2. *Learning a data pattern* – specialization stripped the control flow so data parts can be easily analysed and for example 'interleaving' pattern can be found. If no pattern found, a mapping of input to output mentioned in feedback is used.

3. *Generalization* of the learned model – the conjectured model is stored to the hierarchy after the task instance is changed (it fails to provide accurate outputs and at least 5 consecutive positive rewards were recorded). It is tried to generalize itself and then with its siblings (models in the hierarchy with the same parent) such that the generalized model is a specialization of their parent. The aim is to have a more accurate 'initial' model for next tasks. Finally, the current trace and related ODFSM is cleared and the phase 1 starts for the new task instance. If the conjectured model fails but the task instance was not learnt, then the phase 1 starts again after throwing the incorrect conjectured model away.

Having models of all learnt tasks with their generalized versions is beneficial because one can analyse and combine them and so next related task can be solved faster.

3 Prototype of the Gradual EFSM Learner

A prototype of the EFSM learner was implemented in Python. It does not cover the entire idea so it is rather a showcase how efficient the learning using EFSM hierarchy can be. I focused on the learning of commands so the first four Micro-tasks were skipped. The 'basic communication' EFSM shown in the middle of Figure 1 is considered as the root of hierarchy. The model is possible to learn from the Micro-tasks 1–5 with an implementation of full idea (that would take me more than spare time of one month).

In the current implementation, each EFSM has two data structures, *words* and *mapping*, and two variables, *output* and y. As the implemented phase 2 can detect only simple patterns like copying, interleaving and reverse order, the Learner solves the Micro-tasks 5.3–9 but should be able to learn even the Micro-tasks 14 and 15 easily. The phase 2 could use neural nets or a pool of function that the Learner could combine. That is a future work.

The Learner accepts two file names as optional arguments. The first is name of a pickled pre-learned hierarchy; if the empty string is given, the Learner starts with just the basic communication EFSM as the root. The hierarchy is stored to 'brainTmp.pkl' after each task instance is learnt. If the second argument is provided, the Learner stores the hierarchy in the DOT language to the file with the given name. The hierarchy can be then visualized in FSMvis (based on visjs.org) that is submitted with the source codes.

4 Conclusion

Learning EFSM hierarchy is suitable to use when regular languages are to learn as the hierarchy provides **accurate models** that are **understandable for people** so that one knows exactly how the machine 'thinks'.

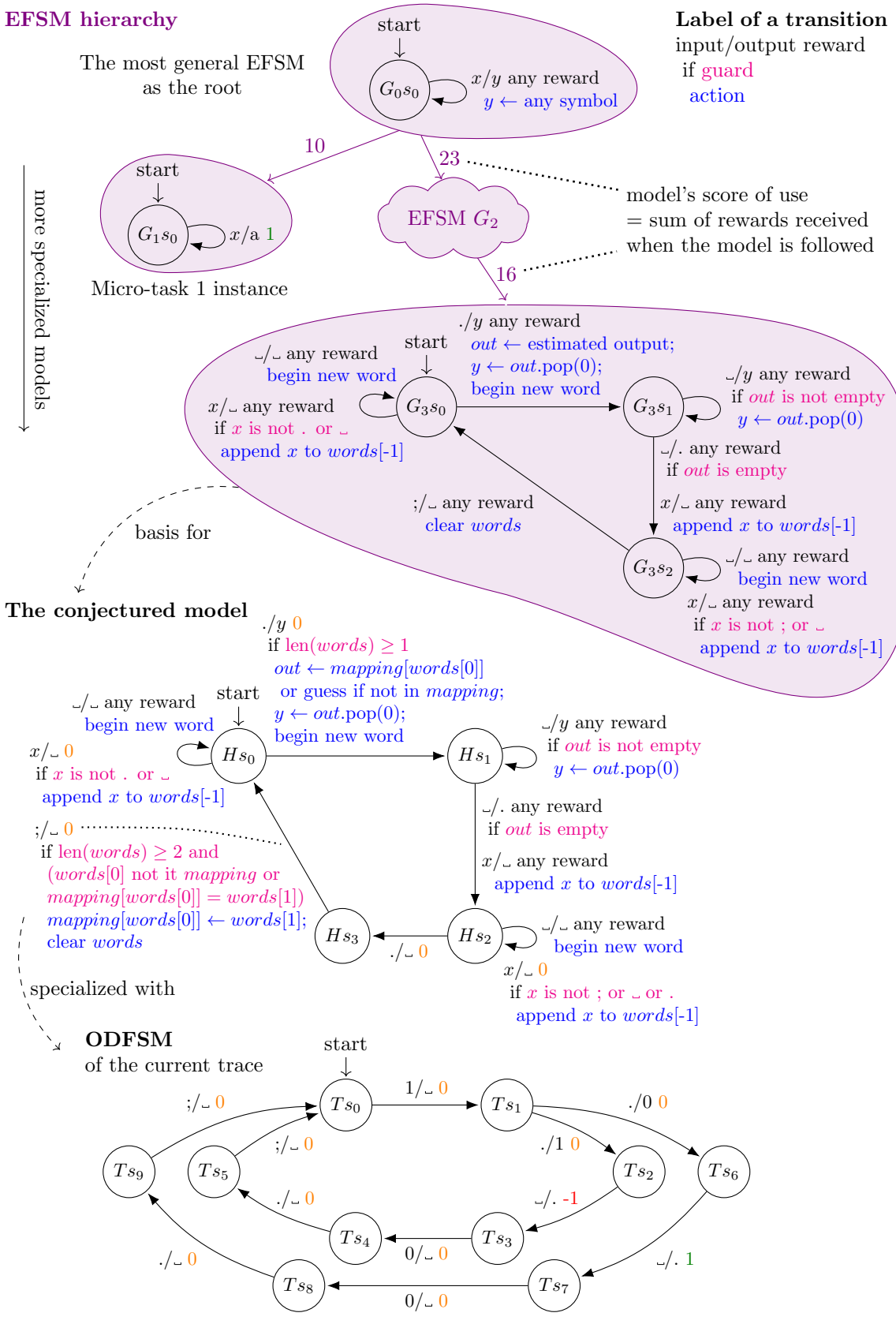

Figure 1: A part of the EFSM hierarchy during learning the Micro-task 5.5