State Machines

The Big Ideas:	• The structure of a program has a significant effect on its modularity • Recursion is expressive • State machines are computation with memory • Output and next state depend on input and current state • A particular kind of state machine is a subclass of SM • A particular state machine is an instance of that subclass • State machines can be used to represent the controls for a system • State machines can be used in analysis and prediction of behaviors for a system

Introduction

Last week, we introduced you to some of the core concepts of 6.01. Our four units are Programming and State Machines, Signals and Systems, Circuits, and Probability and Planning. We focused on programming, in particular the Object Oriented Programming paradigm in Python.

This week, we expand on programming, introducing other programming paradigms and indicating features of Python that tie in to notable programming concepts.

This week, we also introduce state machines. State machines model systems that are functional, but also have memory. State machines are incredibly general, but incredibly powerful, and can be used to model all kinds of systems, as you'll see in future weeks. You can use state machines to control, model, and predict behaviors in systems.

Vocabulary

In order to engage the material, be able to communicate about the topic with others, and in particular ask questions, we encourage familiarity with the following terms:

Theory

- Imperative Programming
- Functional Programming
- Functions as First-Class Objects
- Recursion
- Object Oriented Programming
- Abstraction and Modularity
- State Machine
- State Transition Diagram
- Transition Table
- Cascade, Parallel, Select, Feedback
- Controller
- Plant

Practice

- `class SM`
  - `start`
  - `step`
  - `transduce`
  - `startState`
  - `getNextValues`
- Soar
  - Brain
  - Simulator/Pioneer
  - Sonars
  - `rvel/fvel`
  - `io.SensorInput`

Check Yourself

After this week in 6.01, you should be familiar with the following:

Theory: you should understand:

- OOP and different paradigms
- State Machines: State transition diagrams, transition tables, expression as math equations
- Simple block diagrams

Practice: you should be able to:

- Locate the 6.01 Software Documentation, and use it effectively
- Run a robot using Soar, both in Simulation and Pioneer modes
- Implement a state machine to control robot behavior using a custom `SM`.

6.01SC Introduction to Electrical Engineering and Computer Science

Spring 2011