Upcoming Schedule
- **Now:** Problem Set 3
- **Monday’s class will meet in Rice Hall Bagel Shop Area**
- **Monday, 3 October:** Problem Set 4
- **Wednesday, 12 October:** Exam 1 Due (will be take-home, handed out on Friday, 7 October)

Notes and Questions

What makes a good model?

To model a computer, what are the three things we need to model?

Turing Machine

A *Turing Machine* is an abstract model of a digital computer. It consists of:

- An **infinitely long tape**, divided into squares. (The tape is usually thought of as infinitely long only in one direction, but it is equivalent in power to a tape that is infinitely long in both directions.)
- A finite **alphabet** of symbols. There is a finite set of symbols that can be written into squares on the tape.
- A **tape head** that can read the alphabet symbol on a single square of the tape. For each step, the tape head reads the symbol at the current tape position, and can move one square either left or right.
- A **finite state machine** (see back) that controls the tape head.
Finite State Machine

A *Finite State Machine* is a very simple model of a machine that has a finite amount of memory (unlike the Turing Machine model which has an *infinite* amount of memory since the tape is infinitely long). A Finite State Machine consists of:

- A finite *alphabet* of symbols. There is a finite set of symbols that can be written into squares on the tape.
- A finite set of *states*. Some of the states may be distinguished (for the FSM for a Turing Machine, we typically have a distinguished state called "Halt" where the machine stops if that state is reached).
- A set of *decision rules*. Each rule is of the form \(<state_0, symbol> \rightarrow state_1\). If the machine is currently in \(state_0\) and the next input symbol is \(symbol\), after reading the \(symbol\) the state is now in \(state_1\).

Unlike a Turing Machine, a Finite State Machine can see each input symbol only once. You can think of it as a machine that starts with the input on a finite tape, and process that input from left to right, reading one square at a time.

Design a finite state machine that checks the parity of a binary number. Your machine should end in state \(0\) if the input has an even number of 1’s, and end in state \(1\) if the input has an odd number of 1’s.

Take-Home Problem

Design a Turing Machine that starts with an input tape that starts with a “#”, is followed by a series of “∗” and “♦” symbols, followed by a “#” at the end. The output should be the number of “∗” symbols. A first version should produce the output in unary, leaving the output tape with a sequence of “1” symbols followed by a “#”. For example, if the input tape is #∗♦♦∗♦∗∗♦♦∗♦♦♦ the output tape should be “#11111#”. (A gold-star bonus solution would produce the output in binary notation, instead of unary.)