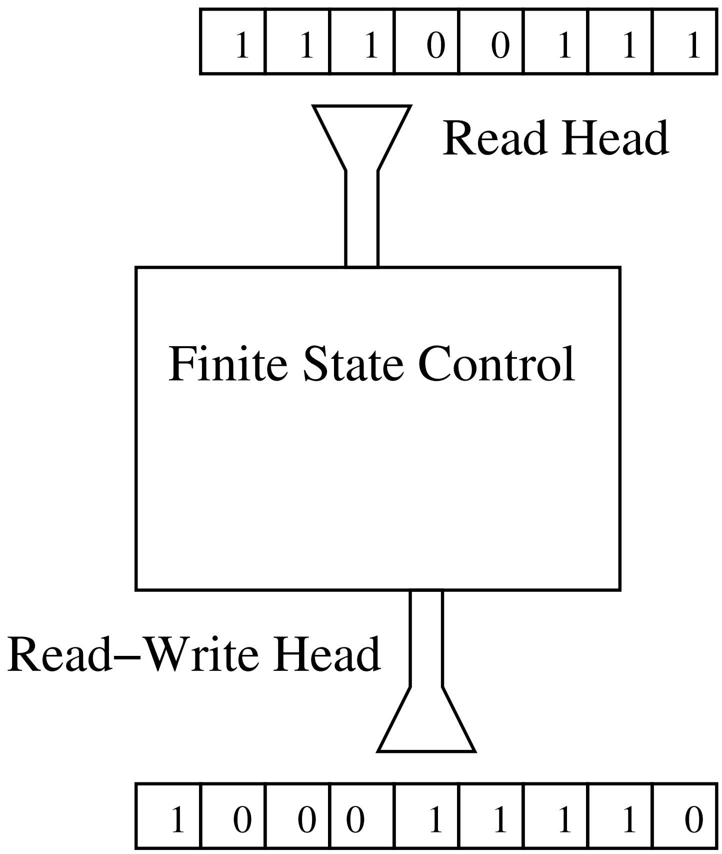
An Informal Introduction to Computability and Computational Complexity

- **Turing Machines** are at the top of the computational hierarchy.



- A TM has an input tape, a finite state controller, and a working tape. The finite state controller can read and write symbols from/to the working tape.

More about UTMs

- A universal TM is a TM that can simulate any other TM.
- A UTM is the most general model of computation. It can match the power of any other computational method.
- Thus, UTM's are equivalent to C programs, java programs, etc.
- TM's are also taken as being equivalent to algorithms.

More about UTMs, continued

- Recursively Enumerable Languages are recognized by UTMs
- However, there exists some languages that no machine can recognize! Why?
- The number of UTMs is countably infinite.
- The number of languages is uncountably infinite, since it is the set of all subsets of a countably infinite set.
- Thus # of Languages $>$ # of machines.

This has some profound and important implications. We can use this to show that some algorithms do not exist.

We will do so via a paradox

Paradoxes

Paradoxes are unavoidable with self-referencing systems.

Examples:

- “I’m lying”
    1. If I’m lying, I’m telling the truth
    2. If I’m telling the truth, I’m lying
- The shortest integer that can’t be described in less than thirteen words.
- Consider the set of all sets that are not members of themselves Should this set be a member of itself?
- etc.

Halting

- One of the problems with UTMs is that sometimes they don't halt. They can get caught in loops.
- Let's see if we can come up with a method for determining if a UTM, with program P and input x will halt.
- This slide and the following is almost directly taken from Randy Wang's lecture on computability: `www.cs.princeton.edu/~rywang/99f126/slides.html`.
- Call this program $\mathrm{HALT}(P, x)$. It takes as input the program P and the output x.
- $\mathrm{HALT}(P, x)$ returns YES if $P(x)$ halts, and NO if it doesn't.
- Will assume that $\mathrm{HALT}(P, x)$ always halts—it does not go into loops.

Halting Problem, continued

Now, let's construct the strange program $\mathrm{XX}(P)$, as follows:

- $\mathrm{XX}(P)$ calls $\mathrm{HALT}(P, P)$.
- $\mathrm{XX}(P)$ halts if $\mathrm{HALT}(P, P)$ outputs NO.
- $\mathrm{XX}(P)$ infinite loops if $\mathrm{HALT}(P, P)$ outputs YES

In other words:

- If $P(P)$ does not halt, $XX(P)$ halts.
- If $P(P)$ halts, $XX(P)$ does not halt.

Let's call XX with *itself* as input

- If $XX(XX)$ does not halt, $XX(XX)$ halts
- If $XX(XX)$ halts, $XX(XX)$ does not halt

Both lead to a contradiction. Therefore, $\mathrm{HALT}(P, x)$ **cannot exist**.

Consequences of Halting Problem

- There does not exist an algorithm to determine if a UTM running program P with input x will halt.
- We say that the Halting problem is **uncomputable** or unsolvable.
- It turns out that many other problems are uncomputable as well.
- Of particular relevance to us: There does not exist an algorithm that will determine the shortest program that will output a given string.
- I.e., there is no general-purpose algorithm for optimal data compression.
- This means that measures of complexity or randomness based on minimal UTM representations are uncomputable.
- More generally, the existence of uncomputable problems means that we'll never be able to find algorithms for everything.

A Very Brief Discussion of Computational Complexity

- The computational complexity of an algorithm is the run time $T(N)$ needed for the algorithm to run, expressed as a function of the size of the problem N.
- The slower $T(N)$ grows with N, the more tractable (and less complex?) the problem is.
- For applications to physics, see, e.g., the work of Machta `www-unix.oit.umass.edu/~machta/`, and Moore, “Computational Complexity in Physics,” `www.santafe.edu/~moore/pubs/nato.html`.
- Computational Complexity is usually concerned with the time scaling of the *worst* cast scenario. The time scaling of the average case may be more relevant. Unlike comp complexity, information theory is concerned with *average* case behavior.
- There has recently been work in applying parallel computational complexity to physical models. This may be more relevant—nature is often parallel.